MACHINE LEARNING BASED DESIGN AND OPTIMIZATION FOR HIGH-PERFORMANCE SEMICONDUCTOR PACKAGING AND SYSTEMS

A Dissertation
Presented to
The Academic Faculty

By

Hakki Mert Torun

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology

December 2020

MACHINE LEARNING BASED DESIGN AND OPTIMIZATION FOR HIGH-PERFORMANCE SEMICONDUCTOR PACKAGING AND SYSTEMS

Thesis committee:

Prof. Madhavan Swaminathan, Advisor
School of Electrical and
Computer Engineering
Georgia Institute of Technology

Prof. Sung-Kyu Lim
School of Electrical and
Computer Engineering
Georgia Institute of Technology

Prof. Saibal Mukhopadhyay
School of Electrical and
Computer Engineering
Georgia Institute of Technology

Prof. Surya R. Kalidindi
School of Mechanical Engineering
Georgia Institute of Technology

Prof. Arijit Raychowdhury
School of Electrical and
Computer Engineering
Georgia Institute of Technology

Date approved: October 16, 2020

To my loving spouse, *Su*, and parents, *Hayri* and *Hatice*,

without whom none of my success would be possible.

ACKNOWLEDGMENTS

The path to a doctoral degree is known to be long and full of hardship, but I was fortunate enough to be surrounded by many amazing individuals that made this journey also full of joy. I would like to express my sincere gratitude to all of them for their constant support, guidance, help and memories that I will cherish for a lifetime.

First and foremost, I would like to thank my advisor, Prof. Madhavan Swaminathan, for giving me the opportunity and believing in me to pursue a PhD in his research group. Throughout my time at Georgia Tech, he patiently guided me in the right direction, shared invaluable technical insights and continuously encouraged me to be the best version myself. His enthusiasm on innovation made me enjoy research more than I could ever imagine and motivated me to overcome its challenges. He has been, to the every definition of the word, the perfect advisor and role model to me. I cannot remember how many times I walked into his office fully confused and overwhelmed with many problems, both technical and personal, and walking out knowing exactly what the right thing to do is. I could not ask for a better PhD experience and for that, I will always be grateful to him.

I also would like to thank my committee members: Prof. Saibal Mukhopadhyay, Prof. Arijit Raychowdhury, Prof. Sung-Kyu Lim and Prof. Surya R. Kalidindi, for their time and valuable feedback to improve my research.

In addition, I would like thank Prof. Gregory Durgin and Dr. Francesco Amato, for mentoring me in my first semester at Georgia Tech and giving me the chance to work on very exciting projects.

I also would like to thank the support and feedback that I received from industry members of Center for Advanced Electronics through Machine Learning (CAEML). Especially to Dr. Jose Hejase and Dr. Dale Becker from IBM, Dr. Ahmet Durgun and Dr. Kemal Aygun from Intel, for their invaluable insights and feedback to constantly improve my research to address real-world problems.

Many of the results in this thesis would not have been possible without the help and motivation of my friends in the lab: Dr. Huan Yu, Dr. Colin Pardue, Sridhar Sivapurapu, Osama Bhatti, Mutee ur Rehman, Serhat Erdogan, Claduio Alvarez, Seunghyup Han, Lakshmi Narasimha, Xiaofan Jia, Majid Ahadi, Nahid Amoli and Kai-Qi Huang. Thank you very much for all the joyful memories.

A heart full of thanks goes to my friends I met in Atlanta: Baki, Berkay, Sezen, Emre, Beren, Bige, Gozde. I truly appreciate your constant support, all the fun memories and trips we took together.

I was fortunate enough to have the support and company of my friends since childhood, scattered around the world pursuing degrees of their own: Alp, Denizhan, Metin, Tolga and Baris. Thank you very much for being on my side for as long as I can remember and sparing a piece of your mind whenever I asked.

I owe my deepest gratitude to my family. My father, Hayri, and my mother, Hatice, have been tirelessly guiding and supporting me more than I can describe with any words. All of my accomplishments would not be even remotely possible without their unconditional love. The same goes for my brother and his wife, Berk and Yasemin, who constantly helped me at every step.

Finally, I would like express my deepest appreciation to my loving spouse, Su. Without her devotion, constant motivation and endless love, I would be lost in this journey. Thank you very much for giving me a purpose in life, your patience to put up with my most stressed times and making me want to be a better person. Her love and support are the basis of all my accomplishments, including this thesis.

This thesis was funded in part by the NSF under Grant NO CNS 16-24811 and the industry members of Center for Advanced Electronics Through Machine Learning (CAEML), DARPA CHIPS project under Award N00014-17-1-2950, 3D Packaging Research Center (PRC) at Georgia Tech, and through internships at POWER Series Hardware Group at IBM, Austin, TX in Summer 2018 and Assembly and Test Technology Development department at Intel, Chandler, AZ in Summer 2019.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ADD-GP	Additive Gaussian Processes
ARD	Automatic Relevance Determination
BNN	Bayesian Neural Network
BO	Bayesian Optimization
CEL	Causality Enforcement Layer
CNN	Convolutional Neural Networks
DPT-BO	Bayesian Optimization With Deep Partitioning Tree
DSE	Design Space Exploration
EDA	Electronic Design Automation
EI	Expected Improvement
FFNN	Feed Forward Neural Network
GP	Gaussian Process
IVR	Integrated Voltage Regulators
MAP	Maximum A Posteriori Estimation
MCMC	Markov Chain Monte Carlo
NMSE	Normalized Mean Squared Error
PDN	Power Delivery Network
PEL	Passivity Enforcement Layer
PI	Probability Of Improvement
PSD	Positive Semi-definite
RCS	Radar Cross-section
S-TCNN	Spectral Transposed Convolutional Neural Network
SI & PI	Signal And Power Integrity
SoC	System-on-chip
TSBO	Two-Stage Bayesian Optimization
UCB	Upper Confidence Bound
UQ	Uncertainty Quantification

SUMMARY

The thesis develops machine learning based models and algorithms that enable accurate system-level optimization, design space exploration (DSE) and uncertainty quantification (UQ) for high-performance microelectronics packaging design for applications including high-speed chip-to-chip signaling, power delivery, embedded passives for wireless power transfer and sub-THz wireless communication, and power modules for electric vehicles.

Increasing the performance of systems integrated in package are often realized through high-density integration of different modules and increasing the operating frequency. However, these raise new challenges to design optimization due to increased non-linearity and higher-order interactions between the various design parameters of the system. Optimization of such systems correspond to finding the minimum or maximum of a high-dimensional, non-convex function in the black-box setting where the function queries correspond to CPU intensive multi-physics simulations such as full-wave electromagnetic and/or computational fluid dynamics based analysis. Conventional methods either get stuck in a local optima, or require numerous system simulations that take impractical computational time and resources.

For the problems related to DSE and UQ, an accurate data-based model of the system is often desired to run a comprehensive parametric analysis. Conventional techniques to derive such models either suffer from low accuracy due to high-dimensionality, high non-linearity or scarce data. Further, it is not possible to quantify the quality of conventional models to assess the reliability of their predictions at inference time.

To address these challenges, the first part of thesis develops learning based models based on neural networks and Gaussian Processes to be used for DSE and UQ with the goal of learning the mapping from high-dimensional control parameters of the system to its performance metrics, while providing confidence bounds around the predictions to assess prediction quality. The second part of this thesis develops new domain-specific single-

and multi-objective Bayesian optimization based algorithms that are suited to packaging related problems and enable convergence to global optima in minimal number of system simulations. The performance of the newly developed methods in both parts are evaluated on emerging design applications and compared to existing state-of-the-art techniques.

CHAPTER 1
INTRODUCTION

1.1 Motivation

Semiconductor industry has seen a rapid advancement over the last two decades as the number of transistors in a functional chip increased based on Moore's Law. Currently, advanced manufacturing processes allow building system-on-chip (SoC) devices at 5 nm nodes. When combined with techniques such as 3-D die stacking, advanced packaging through heterogeneous integration and fine-pitch interconnect fabrics, it is possible to dramatically increase the integration density to miniaturize multi-functional systems as illustrated Figure 1.1.

In addition to miniaturization, there is a constant trend to increase the performance of microelectronic systems. Applications such as artificial intelligence, high-performance computing, electric vehicles, mobile devices and smart cities continuously drive semiconductor industry to build systems with higher-computational power. This leads to a demand to improve chip-to-chip and wireless communication bandwidth and increase operation frequencies while reducing power consumption to maintain a certain energy efficiency.

Building high-performance and highly integrated systems inevitably increase the design complexity, which introduces significant challenges to deriving a modeling framework to rely on during the design cycle. At a very high level, current modeling frameworks fall into two categories, namely accurate but slow models and fast but inaccurate models. The former utilize electronic design automation (EDA) tools such as 3-D full-wave electromagnetic (EM) solvers and SPICE to characterize electrical performance of a given design, computational fluid dynamics (CFD) for thermal analysis and structural analysis for mechanical performance. The use of complex computer simulations in the design loop, how-

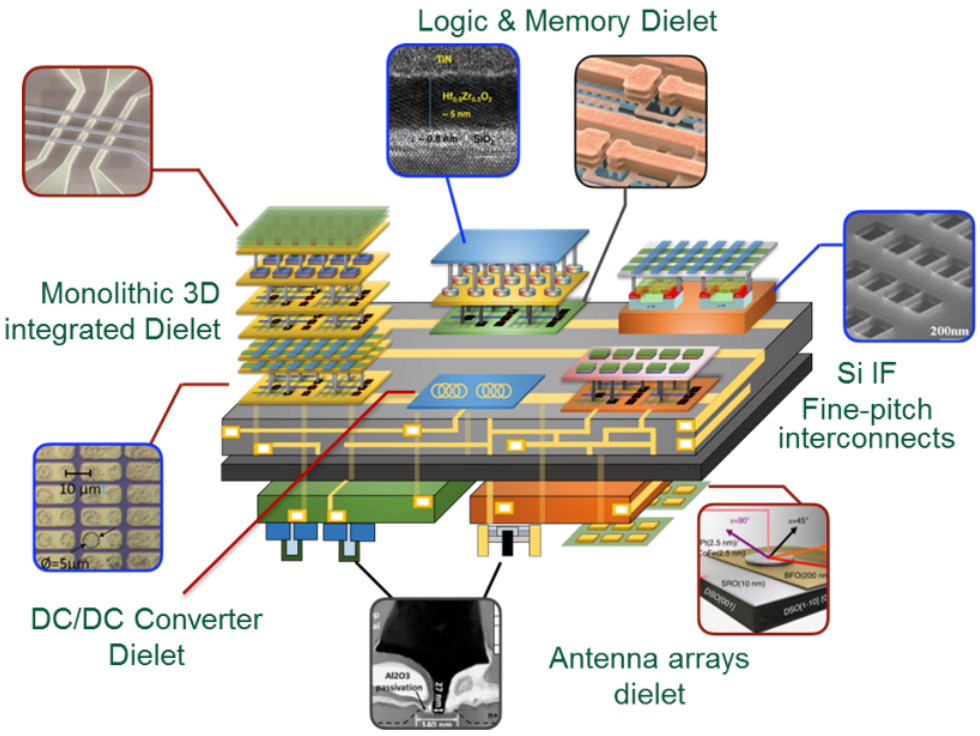


Figure 1.1: Illustration of high-performance and multi-functional enabled through heterogeneous integration and 3-D die stacking. (Courtesy: ASCENT, JUMP)

ever, comes at the cost of increased simulation times and significantly delays the design closure. The use of analytical models addresses the speed of simulations by introducing approximations or assumptions to the modeling framework, but can lead to inaccurate design analysis. Such inaccurate modeling frameworks are one of the major causes of design re-spins due to mistuned design parameters.

An alternative framework is to utilize data-driven models that are derived using machine learning (ML) techniques. Such models can have the same level of accuracy as EDA tools and be as fast as analytical models, potentially getting best of the both worlds. Here, the accurate modeling framework is used to generate a finite amount of training data. This data is then used to train a learning-based model that can accurately predict the outcome of the simulation without actually running it. The derived ML model can then be used to perform rapid design space exploration (DSE) to analyze the effect of different design choices, design optimization to maximize its performance and uncertainty quantification

(UQ) to analyze the effect of process variations on the performance.

Although very promising, conventional ML models have two main limitations: being "data hungry" and "unreliable". The number of training samples required to train a model increases exponentially as the number of design parameters increases. As the increase in design complexity cause an increase in the dimensionality of the design space, building an accurate ML model for complex systems requires significantly more training data. In addition, since most commonly used ML models in EDA are of deterministic nature, there is no principled way to quantify the amount of training data required to maintain a certain predictive accuracy. The problem of unreliability also arises from the deterministic nature of ML models. The underlying assumption behind such deterministic models is that once the training phase is finished, the model is assumed to provide correct predictions for every query even though the query point is not included in the training data. Hence, there is no way to quantify if these predictions are erroneous or not during the inference time. These two limitations prevent widespread adaptation of ML based modeling frameworks by the semiconductor industry, which points a clear need and motivation to improve ML based modeling techniques to overcome challenges in modeling microelectronic devices.

1.2 Summary of Contributions

The objective in this research is therefore to address the limitations of the current ML based modeling frameworks by integrating *domain-knowledge* into ML models and focusing on *probabilistic* ML models that are based on principles of Bayesian learning. The use of domain-knowledge allows to exploit known structures in the data to increase predictive accuracy of ML models when the data is of limited size and the probabilistic nature of Bayesian learning allows to address the aforementioned unreliability issues. When combined, they provide a promising path to address the two main limitations of existing ML based frameworks used in EDA community. Hence, this dissertation aims to develop Bayesian learning based probabilistic models and algorithms that capture domain-

knowledge to enable rapid yet efficient system-level design optimization, design space exploration and uncertainty quantification for high-performance microelectronic packaging design, while focusing on emerging applications such as high-speed signaling, power delivery, sub-THz passives and power module cards for electric vehicles. More specifically, contributions of thesis can be summarized as:

1. A novel convolutional neural network architecture to parameterize frequency responses of electronic systems to be used for rapid design space exploration. The focus here is to enable predicting the whole frequency response in a large frequency-bandwidth. We then introduce a technique establish confidence bounds around the predicted frequency response.

2. A novel physically-consistent neural network architecture to create a parametric model of broadband S-Parameters. Several new techniques are introduced to ensure neural network predicted S-Parameters and the associated confidence bounds satisfy causality and passivity conditions such that they can be directly used in time-domain analysis.

3. A novel Bayesian based global optimization algorithm to specifically address challenges related to optimization of electronic systems. In particular, the goal is to develop a method that enable accurate and efficient simulation based design optimization to find optimal design parameters within a large design space.

4. A novel high-dimensional Bayesian optimization (BO) algorithm to perform global optimization with 30+ design parameters in the blackbox setting. The focus here is on addressing the statistical and algorithmic challenges related to high-dimensional optimization by introducing appropriate assumptions for high-frequency electronic design.

5. A novel mixed-variable multi-objective BO method to perform optimization in a mixed-variable domain consisting of multiple continuous and categorical parameters.

Here, the focus is on multi-physics optimization of a power module package design with the goal of simultaneously finding optimal package architecture, materials used at each layer of package, and its physical layout.

6. A novel Bayesian Active Learning based model building technique to perform a complete uncertainty quantification of electronic systems to account for processes variations. A new method is introduced to minimize the number of training data samples required to build a Bayesian model to obtain the worst-case design performance, obtain its probability density function (PDF), and perform sensitivity analysis. Since the focus is on minimizing the amount of training data required, a methodology to establish well-calibrated confidence bounds around the model predictions is presented.

1.3 Outline of the Dissertation

The rest of this thesis is organized as follows: Chapter 2 provides a brief background on data-driven modeling, Bayesian learning, neural networks, Gaussian Processes and Bayesian optimization (BO), Chapter 3 presents a transposed convolutional network architecture to create predictive models of frequency responses, Chapter 4 presents a physically consistent neural network architecture to ensure neural network predicted broadband S-Parameters are physically consistent, Chapter 5 presents the BO technique developed specifically for electronic design optimization and applies it to two emerging design applications, Chapter 6 presents a high-dimensional BO method developed for high-frequency design optimization and its application to three design problems, Chapter 7 presents a mixed-variable GP model that is used for single- and multi-objective optimization and its application to power module package optimization, Chapter 8 presents a Bayesian Active Learning framework to perform a complete uncertainty quantification of electronic systems and its application to high-speed channel signaling, followed by summary and future work in Chapter 9.

CHAPTER 2

BACKGROUND

Throughout this thesis, we will build on concepts such as data-driven modeling, neural networks, Bayesian learning and regression, Gaussian Process and Bayesian optimization. In this chapter, we provide a background on these core concepts that will be utilized in subsequent chapters.

2.1 Data-Driven Modeling

Data-driven modeling in the context of hardware design most commonly refers to replacing an accurate but slow simulation framework with a learned model that is very fast to evaluate, while providing the same level of accuracy as the actual simulation framework. Let $\mathbf{X} = \{x_1, \cdots, x_D\}$ represent the design parameters of interest as a D-dimensional vector and y be the output of interest. The simulation framework can then be interpreted as a non-linear "blackbox" function that takes the design parameters as input and provides the output of interest, i.e. $y = f(\mathbf{X})$. At a high-level, the goal of data-driven modeling in the deterministic setting is non-linear regression. That is, finding a compact function with a set of modifiable parameters, $g(\mathbf{X}; \theta)$, such that once the correct parameters are found, it can be used to replace the actual simulation framework.

To derive such a model, a set of training data, $(\mathbf{X}_T, \mathbf{Y}_T) = \{(\mathbf{X}_1, y_1), \cdots, (\mathbf{X}_T, y_T)\}$, consisting of T samples is first collected using the actual simulation framework. The training data is then used to determine the model parameters that minimize the error between model and training data. This process is called as training the model and can be written as

$$\widetilde{\theta} = \arg\min_{\theta} \sum_{t=1}^{T} \big(g(\mathbf{X}_t; \theta) - y_t\big)^2 \tag{2.1}$$

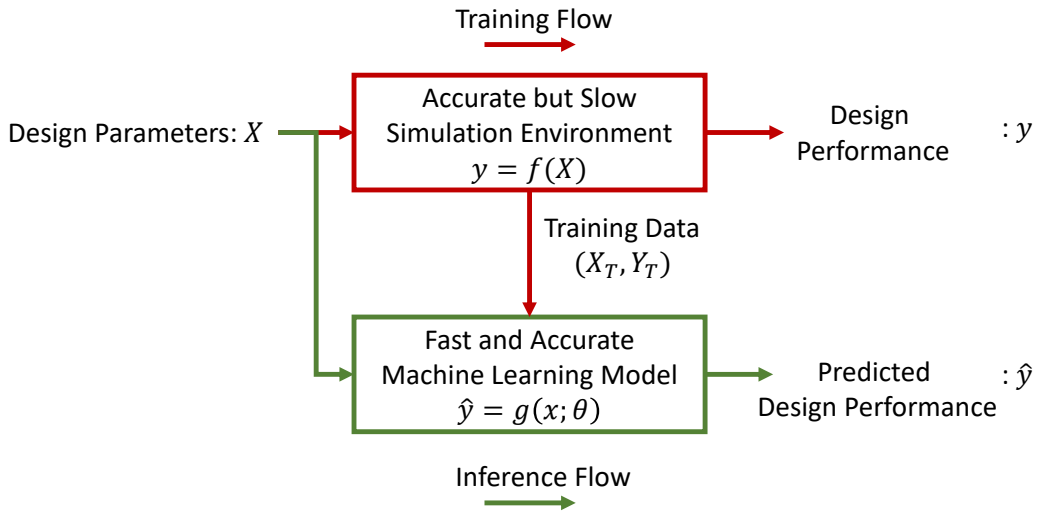


Figure 2.1: Fundamentals of data-driven modeling.

where $\widetilde{\theta}$ is the optimal model parameters such that $g(\mathbf{X}; \widetilde{\theta}) \approx f(\mathbf{X})$. After the training is completed, the model is verified on an independent set of validation samples to assess its predictive accuracy. If the validation accuracy is high enough for a given application, the model derivation process is said to be completed. The derived model can then be used infer the output that correspond to any input vector, effectively eliminating the simulation framework from the design loop as in Figure 2.1. In other words, one can use the derived-model instead of the actual simulation framework to perform rapid and accurate parametric analysis to explore the design space, quantify the effect of process variations and perform design optimization.

2.2 Neural Networks

One of the most well-known and commonly used models in data-driven modeling is called neural networks (NN), a biologically inspired computational paradigm that dates back to 1943. Here, the goal is to try and mimic the neuron-based decision making system of the human brain. Therefore, the building block of a NN is called a neuron as shown in Figure 2.2(a).

A neuron takes a vector of inputs $\mathbf{X} = (x_1, \cdots, x_n)$, constructs their weighted sum

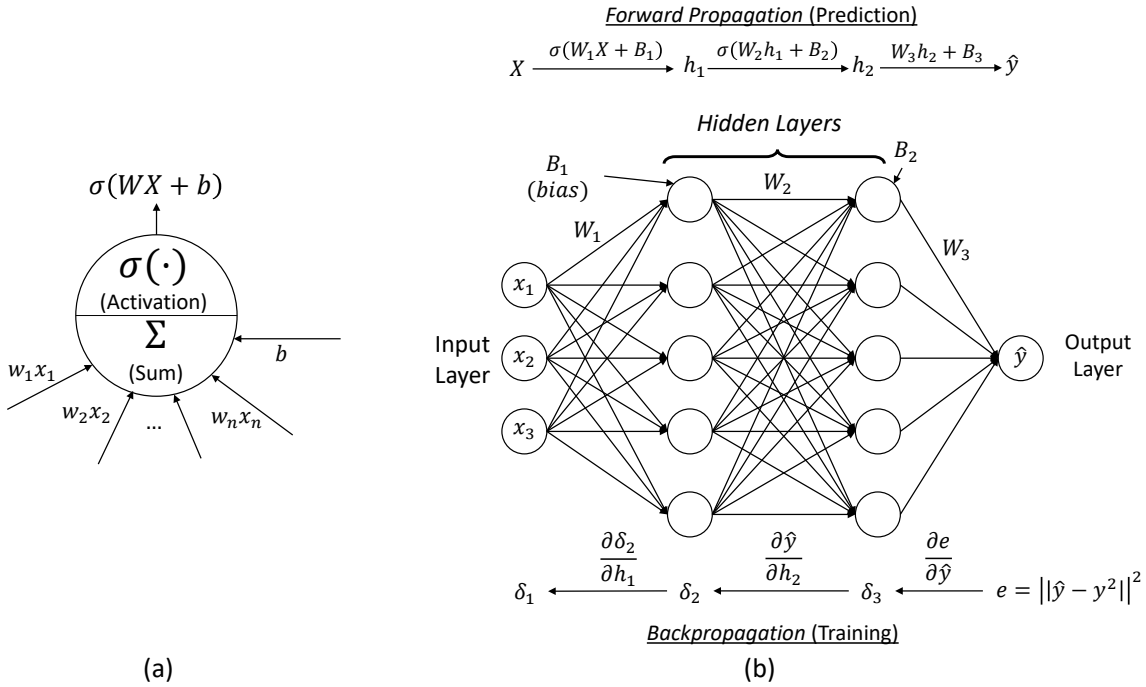


Figure 2.2: (a) Single Neuron and (b) Feed Forward Neural Network (FFNN).

$\mathbf{WX} = (w_1x_1, \cdots, w_nx_n)$ and adds a bias $(\mathbf{b})$ to generate $\mathbf{WX} + \mathbf{b}$, where $\mathbf{b}$ is similar to an intercept term in conventional linear regression. This is then passed through a non-linear activation function to get $\sigma(\mathbf{WX} + \mathbf{b})$, which represents the output of this single neuron as $\sigma(\mathbf{WX} + \mathbf{b})$. The purpose of this activation function is to introduce non-linearity to the overall framework.

A NN connects several of these neurons as shown in Figure 2.2(b). A typical architecture consists of an input layer, multiple hidden layers and an output layer. The input layer consist of the input variables (design parameters) that we want to map to an output. The purpose of the hidden layers is to capture non-obvious interactions between the overall input-output relationship. We use multiple neurons in each hidden layer and connect each one to all the other neurons in the subsequent layers, where each connection describes a different interaction pattern. As the number of hidden layers increase for capturing more complex patterns in data, the NN becomes a Deep Neural Network (DNN).

The output of the output layer, i.e. NN generated output, is then compared with the actual output (training data) to calculate error (e). This error is then minimized by adjusting

the weights in each layer through their gradients, and this process is known as training the NN. To make the training procedure computationally efficient, the gradients are calculated through chain rule of derivatives, i.e. the gradient of weights in a particular layer is *back-propagated* to the previous layer as shown in Figure 2.2(b). Since the data moves from input to output layer in a single direction, we call this as a feed forward neural network (FFNN). FFNNs and many backpropagation based training algorithms are widely available as plug-and-play modules in many programming languages such as Python and MATLAB [1, 2]. It is important to note that the type of activation function, number of neurons per layer and number of layers in the NN, collectively referred to as the *hyperparameters* of the NN, can vary based on the data sets. These hyperparameters should be optimized for each modeling task with the goal of minimizing the validation error.

In the domain of semiconductor packaging, FFNNs are widely used for many applications including high-speed interconnect [3] and embedded passive analysis [4], predicting target impedance violations in a power delivery network (PDN) [5], eye diagram modeling [6, 7, 8] and current prediction at package pins [9].

It should be noted that there are many other NN architectures that are useful for packaging design. For instance, recurrent neural networks (RNN) are commonly used for time-domain problems such as modeling of transient non-linear behavior of I/O buffers [10, 11, 12, 13]. Another type of architecture, called as convolutional networks, is useful for frequency-domain problems as shown in Chapter 3 of this thesis.

2.3 Bayesian Learning

Although promising, there are two major problems with the deterministic data-driven modeling and NNs: 1) How to know if the derived model $g(x, \widetilde{\theta})$ is accurate enough? and 2) How many training samples are required to obtain a certain level of predictive accuracy? In fact, the only check done when deriving the predictive model is to evaluate its performance on a small validation set, and there is no other way to further assess the quality of the model

at inference time.

The ideal model to replace an accurate but slow simulation framework should be reliable, not too complex to fit random fluctuations - noise - in the dataset but also not too simplistic to miss complicated trends. Bayesian learning techniques provide for an elegant framework to derive a reliable model that encourages models with the right complexity.

Consider that we have a training dataset of T samples, $(\mathbf{X}_T, \mathbf{Y}_T)$, and we want to perform non-linear regression. In the Bayesian learning framework, instead of trying to learn a deterministic function, $f : X \rightarrow y$, we aim to learn a probabilistic model, $p(f|\mathbf{X})$, where each prediction now comes with a confidence bound that can be used to quantify the model accuracy. There are two main approaches to learn such a probabilistic model. The first is to place a prior on the model parameters and learn $p(\theta|\mathbf{X}_T, \mathbf{Y}_T)$. The goal here is to learn all possible model parameter combinations that can be associated with the current model and dataset. This approach will be explored later in Chapters 3-4 of this thesis where we present how to convert deterministic neural networks into a Bayesian model to obtain confidence bounds around predictions.

In this chapter, we focus on the second way to obtain a Bayesian model, that is placing a prior directly on the function to be learned. To achieve this, we first choose a model structure that reflects our prior beliefs about the underlying function, and then combine with the dataset to obtain a posterior distribution. The framework is based on well-known Bayes' rule, written as

$$p(f|\mathbf{X}_T, \mathbf{Y}_T, \theta) \propto p(\mathbf{Y}_T|f, \mathbf{X}_T, \theta)p(f|\mathbf{X}_T, \theta) \tag{2.2}$$

where $\mathbf{Y}_T = f(\mathbf{X}_T) + \epsilon$ are the noisy observations of the function to be modeled, θ are the parameters of the model to be learned, $p(f|\mathbf{X}_T, \theta)$ is the prior distribution conditioned on model parameters, $p(\mathbf{Y}_T|f, \mathbf{X}_T, \theta)$ is the likelihood that quantifies how likely the observed data can be generated from our prior and $p(f|\mathbf{X}_T, \mathbf{Y}_T, \theta)$ is the posterior that combines

the data with our beliefs. For the purposes of non-linear regression on hardware design problems, observations will generally be made through deterministic multi-physics simulations. However, it is useful to consider the noisy setting and set $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ to account for possible numerical errors involved in the simulation framework, e.g. small meshing errors involved in full-wave electromagnetic (EM) simulations utilizing adaptive meshing techniques.

Similar to the deterministic setting, the model parameters needs to be learned to adapt the prior to the available data. This can be done using maximum a posteriori estimation (MAP) of θ using a gradient ascent based optimization as

$$\widetilde{\theta} = \arg\min_{\theta} \prod_{t=1}^{T} p(\mathbf{y}_t | f, \mathbf{x}_T, \theta) p(\theta) \tag{2.3}$$

or equivalently in a summation form for convenience as,

$$\widetilde{\theta} = \arg\min_{\theta} \sum_{t=1}^{T} \big(\log(p(\mathbf{y}_t | f, \mathbf{x}_T, \theta) + \log(p(\theta)) \big) \tag{2.4}$$

where $\log(\cdot)$ is the natural logarithm operator and $p(\theta)$ is the prior on the model parameters. The $p(\theta)$ term in Equation 2.4 allows to incorporate additional prior knowledge to the learning problem. For instance, if the underlying data to be learned is believed to come from a highly non-linear function, one can choose a prior to favor higher-order functions. If there are no preference or prior knowledge, $p(\theta)$ can be chosen as a uniform distribution and be omitted from the MAP framework in Equation 2.4.

Once the MAP estimate of θ is obtained, we can predict the distribution of the output of interest at a new input configuration, $\mathbf{x}^*$, through conditioning on the observed data as

$$p(y^* | x^*, \mathbf{X}_T, \mathbf{Y}_T, \widetilde{\theta}) = \frac{p(y^*, \mathbf{Y}_T | x^*, \mathbf{X}_T, \widetilde{\theta})}{p(Y_T | X_T, \widetilde{\theta})} \tag{2.5}$$

where $p(y^* | x^*, \mathbf{X}_T, \mathbf{Y}_T, \widetilde{\theta})$ is the predicted distribution of $y^* = f(x^*)$, $p(y^*, \mathbf{Y}_T | x^*, \mathbf{X}_T, \widetilde{\theta})$

is the joint distribution of observed data and the prediction and $p(Y_T|X_T, \widetilde{\theta})$ is the likelihood. The mean of $p(y^*|x^*, \mathbf{X}_T, \mathbf{Y}_T, \widetilde{\theta})$ will now serve as our final prediction at x^* and its standard deviation will be used to quantify the uncertainty around the prediction.

2.4 Gaussian Process

The question remains to be answered is how do we choose the prior distribution over the functions. For DSE, UQ and design optimization problems in hardware design, we need to collect a new dataset for each problem through computationally intensive multi-physics simulations. This significantly limits the size of the dataset to be used for deriving the predictive model. Hence, the prior we choose for Bayesian modeling needs to work well in the scarce data regime and should be able capture complex input-output relationships. In other words, the model needs to be flexible enough to learn arbitrarily non-linear functions, provide reasonable predictive accuracy with a few training examples, reliable confidence bounds around the predictions and be computationally efficient to calculate predictions and other statistical metrics of interest.

One such prior over functions is called a Gaussian Process (GP), which is a class of non-parametric Bayesian models [14]. GP is the extension of standard multivariate Gaussian distribution to infinitely many variables, any finite number of which having a joint Gaussian distribution. As such, it is completely defined by two quantities, a mean function and a covariance matrix. The prior is then defined as

$$f(x) \sim \mathcal{N}\left(\mu(\mathbf{X}_T), K_{\mathbf{X}_T}\right) \tag{2.6}$$

where $\mu(\mathbf{X}_T)$ is the mean function and $K_{\mathbf{X}_T}$ is the covariance matrix evaluated at training inputs. The mean function introduces a strong parametric bias to otherwise fully non-parametric modeling setting. In many problems, this limits the generalization capability of the GP model since predictions far away from the training data will converge to the

prespecified parametric mean function [15]. Although this can be useful for particular problems, for the purposes of general non-linear regression, it is common to use a constant mean function [16], i.e., $\mu(x) = m$.

The covariance matrix is constructed by a kernel function, $k(\mathbf{x}_i, \mathbf{x}_j)$, as follows

$$K_{\mathbf{X}_T} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_t, \mathbf{x}_1) & \cdots & k(x_t, \mathbf{x}_t) \end{bmatrix} \tag{2.7}$$

As the mean function is set to a constant, the covariance matrix, thereby the covariance function, solely determines the types interactions that the GP model can capture. Domain knowledge of the underlying function can be incorporated here by choosing a descriptive kernel function. For instance, the popular squared exponential (SE) kernel is used for infinitely differentiable smooth functions, whereas Matern type kernels are used when the function is believed to be less smooth. A commonly used kernel is Matern 5/2 function with automatic relevance determination (ARD) [16], given as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2(1 + \sqrt{5}r + \frac{5}{3}r^2)e^{-\sqrt{5}r} \tag{2.8}$$

where

$$r = \left(\sum_{d=1}^{D} \frac{(x_{i,d} - x_{j,d})^2}{\lambda_d^2} \right)^{1/2}$$

and $x_{\cdot,d}$ is the d^{th} dimension of input parameter vector, λ_d is the lengthscale of each input parameter and σ_f is the scaling constant. The parameters of this kernel function, λ_d and σ_f, along with noise variance σ_n^2 in Equation 2.2, are collectively referred as the hyperparameters of the GP model, θ.

Once the mean and kernel function is set, the training of the GP is done to find these hyperparameters using the MAP estimation in Equation 2.4. For a uniform prior over θ, the

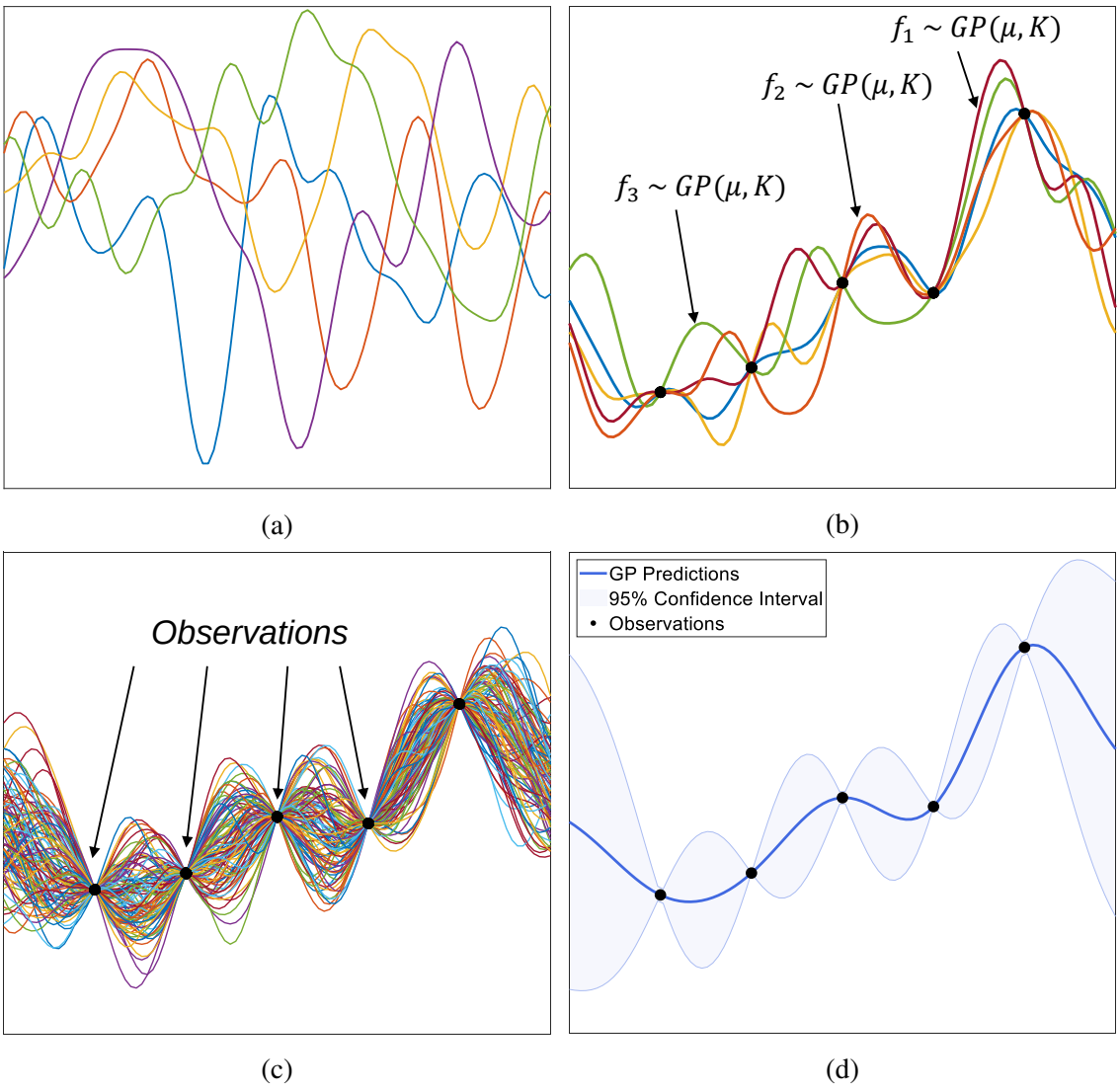


Figure 2.3: Demonstration of sampling functions from a GP using a few observations. (a) Samples from the GP prior without conditioning on the observed data. (b) A few samples from the GP posterior obtained by conditioning on a few observations. (c) Large number of samples to show the general trend. (d) Mean and 95% confidence bound of infinitely many samples calculated using Equation 2.13 and Equation 2.14.

unnormalized log-likelihood of a GP can be written in closed form as

$$\log p(\mathbf{Y}_T|\mathbf{X}_T,\theta) = \underbrace{-\frac{1}{2}\mathbf{Y}_T^T(K_{X_T}+\sigma_n^2 I)^{-1}\mathbf{Y}_T}_{\text{Encourage model accuracy}} \quad - \quad \underbrace{\frac{1}{2}\log|K_{\mathbf{X}_T}+\sigma_n^2 I|}_{\text{Limit model capacity}} \tag{2.9}$$

where I is the identity matrix of size T and $(\cdot)^T$ is the transpose operator. The gradient of the log-likelihood function can also be computed as

$$\frac{\partial \log p(\mathbf{Y}_T|\mathbf{X}_T,\theta)}{\partial \theta_i} = -\frac{1}{2}\text{tr}\left(K^{-1}\frac{\partial K}{\partial \theta_i}\right) + \frac{1}{2}\mathbf{Y}_T^T K^{-1}\frac{\partial K}{\partial \theta_i}\mathbf{Y}_T \tag{2.10}$$

where we used $K = K_{X_T} + \sigma_n^2 I$.

Examining the structure of the likelihood in Equation 2.9 reveals how GP encourages learning the "right model" discussed in previous section. The first part of the log-likelihood encourages exactly fitting to the observed data as in the deterministic setting in Equation 2.1 without considering the complexity of the final model. The second part directly contradicts with the first part and encourages simple models without considering the observations. The optimal θ then becomes the point that balances model accuracy and simplicity. This breakdown shows the underlying idea in Bayesian learning, that is, we want the model that has just the right amount of complexity to avoid both overfitting and underfitting. Although some deterministic models introduce so-called regularization terms to the loss function to limit the model capacity, often times, it is hard to determine the severity of the regularization. In the Bayesian learning framework utilized in GPs, there is no need to fine-tune regularization term as it appears naturally as part of the likelihood function.

Once the training is done, the GP model can now be used to make predictions to be utilized for DSE, UQ and/or design optimization. To make inference using the trained model at M test points, $\mathbf{x}^* \in \mathbb{R}^{M\times D}$, we represent the training and test data to have a joint

block Gaussian distribution as

$$p(y^*, Y_T | x^*, X_T, \theta) = \mathcal{N}\left(\begin{bmatrix}\mu(X_T)\\ \mu(x^*)\end{bmatrix}, \begin{bmatrix}K_{X_T} & K_{X_T,x^*}\\ K_{X_T,x^*}^T & K_{x^*,x^*}\end{bmatrix}\right) \tag{2.11}$$

where $K_{\mathbf{X}_T}$ is as in Equation 2.7 and $K_{\mathbf{X}_T,\mathbf{x}^*}$ is the cross-covariance matrix between the training and test points, respectively. To finally get the predictive distribution at test points, we condition on training data as

$$p(y^* | x^*, X_T, Y_T, \theta) = \mathcal{N}\left(y^* |\ \widehat{\mu}_\theta(x^*), \widehat{\sigma}_\theta^2(x^*)\right) \tag{2.12}$$

with

$$\widehat{\mu}_\theta(x^*) = K_{X_T,x^*}^T (K_{X_T} + \sigma_n^2 I)^{-1} Y_T \tag{2.13}$$

$$\widehat{\sigma}_\theta^2(x^*) = K_{x^*,x^*} - K_{X_T,x^*}^T (K_{X_T} + \sigma_n^2 I)^{-1} K_{X_T,x^*} \tag{2.14}$$

where $\widehat{\mu}_\theta(x^*) \in \mathbb{R}^{M\times 1}$ is the posterior mean and $\widehat{\sigma}_\theta^2(x^*) \in \mathbb{R}_+^{M\times M}$ is the posterior covariance matrix calculated using the learned hyperparameters, respectively. One can now sample from the posterior to get the "predicted functions". A few function samples before and after conditioning to the observed data are demonstrated in Figure 2.3b and Figure 2.3b, respectively, showing the form of the prior and possible structures of the underlying non-linear function. If we keep sampling a very large number of functions as in Figure 2.3c, we observe a trend that each function exactly passes through all the observations, but varies in between them. If we sample infinitely many functions, their average becomes the posterior mean in Equation 2.13, hence, we will use the posterior mean to be our predictions at test points without doing infinitely many samples. The variation of infinitely many functions samples is enveloped using the point-wise variance at test points, which is equal to the diagonal of the posterior covariance matrix in Equation 2.14. We will use this to get the

desired confidence bounds around the predictions.

2.4.1 Effect of the Kernel Function

Since the kernel function determines the structure of the functions that can be captured by a GP model, it is worth exploring some possible kernel functions that might be used for a regression problem. A kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ defines how similar two outputs y_i and y_j to each other based on their corresponding input vector x_i and x_j. It is possible to construct a kernel to capture any domain knowledge about the underlying data through constructing the appropriate kernel function. However, this is generally a non-trivial problem since a kernel function has to have two properties [14]:

Symmetricity $\rightarrow$ The covariance matrix whose elements are $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ need to be symmetric.

Positive Semi-Definiteness $\rightarrow$ The covariance matrix needs to be positive semi-definite (PSD). A real $n \times n$ matrix is said to be PSD if $v^T K v \geq 0$ for all $v \in \mathbb{R}^n$. Note that a symmetric and PSD matrix is always invertible since its eigenvalues are non-negative.

Although the symmetricity can be achieved by using $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$, ensuring the resulting covariance matrix to be PSD is challenging. However, we can create new expressive kernels by combining existing kernels in an additive and multiplicative form as

$$\begin{aligned} k_{\text{new}}(\mathbf{x}_i, \mathbf{x}_j) &= k_1(\mathbf{x}_i, \mathbf{x}_j) + k_2(\mathbf{x}_i, \mathbf{x}_j) \\ k_{\text{new}}(\mathbf{x}_i, \mathbf{x}_j) &= k_1(\mathbf{x}_i, \mathbf{x}_j) \times k_2(\mathbf{x}_i, \mathbf{x}_j) \end{aligned} \tag{2.15}$$

where $k_{\text{new}}(\mathbf{x}_i, \mathbf{x}_j)$ is a valid kernel that holds the two properties of symmetricity and PSD. Note that there are other ways to capture other known structures such as changepoints, multiplication by a known function and capturing invariances. The interested readers are referred to [17] for a more detailed overview.

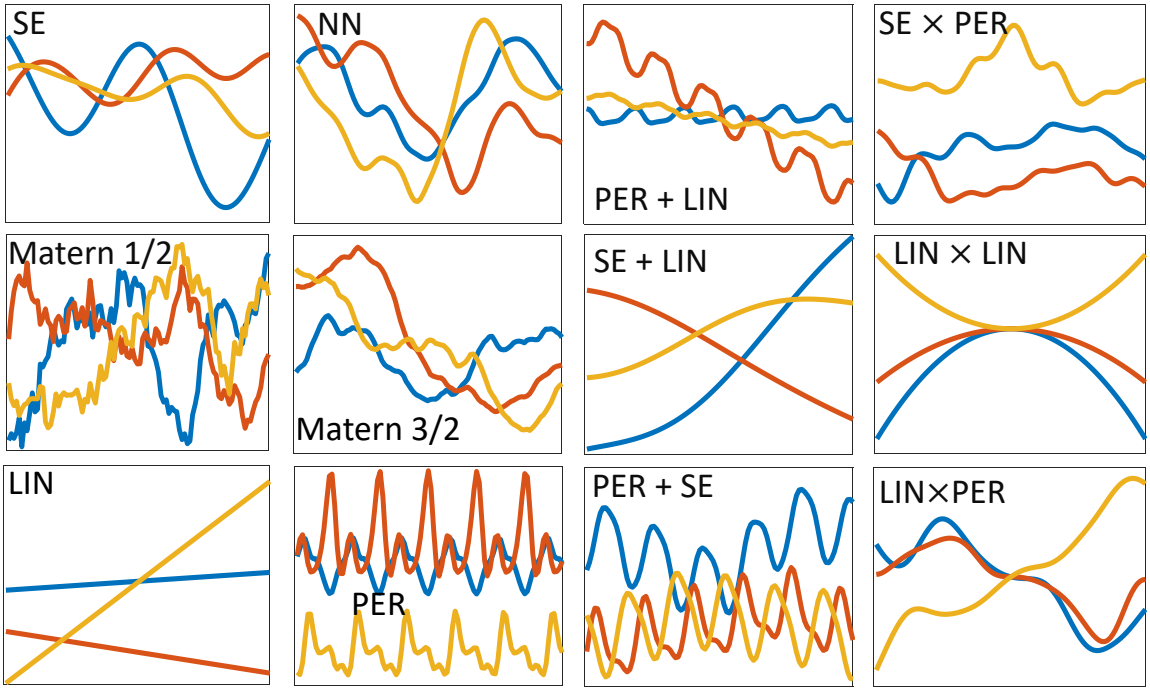


Figure 2.4: Samples from GP prior using different kernel functions. First two columns show standalone kernels, whereas the other two show some of the structures that can be obtained by combining standalone kernels.

Table 2.1: Summary of commonly used kernels and their possible use cases

Name	**Expression** ($k(x_i, x_j) = \cdots$)	**Parameters**	**Use Cases**
Linear (LIN)	$\sigma_0^2 + \sigma_f^2(\mathbf{x}_i - c) \cdot (\mathbf{x}_j - c)$	$\{c, \sigma_0^2, \sigma_f^2\}$	Equivalent to Bayesian linear regression
Squared Exponential (SE)	$\sigma_f^2 \exp(-0.5r^2)$	$\{\lambda_{1,\cdots,d}, \sigma_f^2\}$	Smooth, infinitely differentiable non-linearities
Matern 1/2	$\sigma_f^2 \exp(-r)$	$\{\lambda_{1,\cdots,d}, \sigma_f^2\}$	Highly non-linear data, equivalent to Brownian motion
Matern 3/2	$\sigma_f^2 (1 + \sqrt{3}r)\exp(-\sqrt{3}r)$	$\{\lambda_{1,\cdots,d}, \sigma_f^2\}$	Moderately smooth, non-linear functions
Matern 5/2	$\sigma_f^2 (1 + \sqrt{5}r + 5/3r^2)\exp(-\sqrt{5}r)$	$\{\lambda_{1,\cdots,d}, \sigma_f^2\}$	Functions that are less smooth than SE but more than Matern 3/2
Periodic (PER)	$k_0(u_i, u_j)$ $u_i = [\sin(\pi x_i/p), \cos(\pi x_i/p)]$	$\{\theta_{k_0}, p_{1,\cdots,d}\}$	Captures periodic patterns, useful for time-domain waveforms
Neural Network (NN)	$\sigma_f^2 \sin^{-1}(\frac{x_i^T \Lambda x_j}{\sqrt{(1+x_i^T \Lambda x_i)(1+x_j^T \Lambda x_j)}})$	$\{\lambda, \sigma_f^2\}$	Non-stationary functions, useful to capture sharp changes

$r = \sqrt{\left(\sum_{d=1}^{D} (x_{i,d} - x_{j,d})^2 / \lambda_d^2\right)}$, θ_{k_0} in PER are parameters of k_0, Λ in NN is identity matrix multiplied by $1/\lambda^2$

Some of the existing kernels that can be used as the base to construct new kernels are summarized in Table 2.1 along with their possible use cases. In addition, samples from the GP prior using different kernels are given in Figure 2.4. It can be seen that the different patterns that exist in the function being modeled can be captured by using the appropriate kernel, or by combining two or more kernels. For instance, a standalone PER kernel can be used to model periodic functions and a standalone SE can be used to model smoothly varying functions, whereas a composite PER$\times$SE kernel can be used to model locally periodic functions or PER$+$SE kernel for periodic functions around a smooth trend function.

2.5 Efficient Design Optimization: Bayesian Approach

One of the major benefits of data-driven framework is that it allows to use the derived fast model to be used in a design optimization loop. Once the model derivation is complete, the fast and accurate ML model can be combined with various optimization techniques to find the design parameters that minimize or maximize the objective function. Formally, design optimization can be formulated as a global optimization problem that can be written as

$$\widetilde{x} = \underset{x \in \mathbb{X}^D}{\arg\max}\, f(x) \tag{2.16}$$

where $\mathbb{X}^D$ is the D dimensional sample space and $f(x)$ is the unknown objective function. In the black-box setting, the only way to get information about the underlying system $f(x)$ is through querying the function itself. In packaging problems, this function query corresponds to multi-physics simulations of multi-scale architectures, which becomes very CPU intensive. The data-driven framework replaces the problem in Equation 2.16 with

$$\bar{x} = \underset{x \in \mathbb{X}^D}{\arg\max}\, g(x, \widetilde{\theta}) \tag{2.17}$$

where $g(x, \widetilde{\theta})$ is the ML model that is fast to evaluate and $g(x, \widetilde{\theta}) \approx f(x)$ for every $x \in \mathbb{X}^D$. This approach of replacing $f(x)$ with $g(x, \widetilde{\theta})$ for optimization purposes is also known as deterministic surrogate-based optimization and used excessively in the recent literature [18, 19, 20, 21].

However, the deterministic surrogate-based optimization method assumes that the derived model has high-enough accuracy such that the solution found by optimizing the model or the actual simulation framework is said to be very similar to each other, i.e., $\bar{x} \approx \widetilde{x}$. Since there is no such guarantee or practical measures to quantify the deviation $||\bar{x} - \widetilde{x}||$, we cannot ensure if the parameters found are indeed the optimal parameters or not. Furthermore, the dataset we collect to build the model spans the entire design space $\mathbb{X}^D$. Since we do not know where the optimum point lies in the design space, this means that we waste a significant portion of our computational budget to collect training data where $f(x)$ is potentially sub-optimal.

A promising method to overcome these challenges is Bayesian Optimization (BO) based on GPs. Here, the goal is to directly find the global optimum rather than deriving a high-quality predictive model that covers the entire sample space. Instead of collecting the dataset apriori, the BO approach aims to collect the dataset one-by-one in an adaptive manner.

To achieve this, BO combines two components to solve the optimization problem in Equation 2.16, namely a probabilistic model and a sampling strategy. The role of the model is to "guide" the optimization procedure by approximating the underlying objective function to enable calculation of *statistical metrics*. The role of the sampling strategy is to exploit these metrics and act as the "decision making" arm to determine where to query the function. Each time a new sample is collected, the probabilistic model is updated to reflect the newly obtained information. As such, at t^{th} iteration, the model uses all the available data, $X_t, Y_t = \{(x_1, y_1), ..., (x_t, y_t)\}$, to select the next sampling location, x_{t+1}. The function is then queried at $f(x_{t+1})$ to collect the new information and the process is

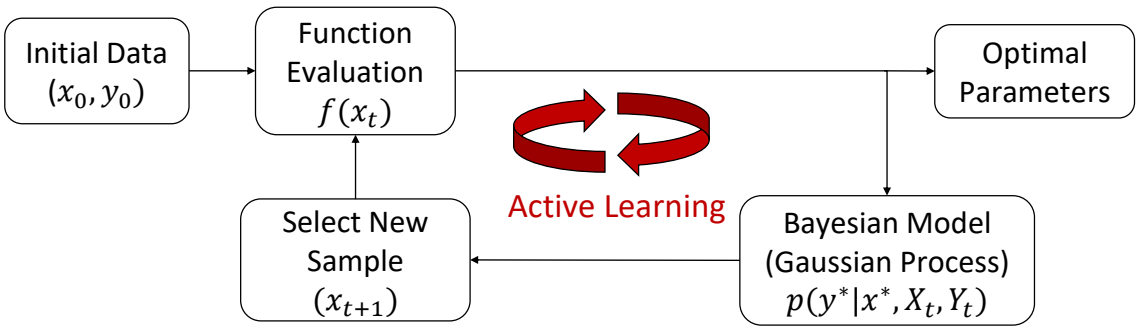


Figure 2.5: High-level summary of BO framework (Modified from [22]).

repeated in a sequential manner until a certain stopping criterion. As x_{t+1} is determined according to the previously obtained information, this is called "active learning" as shown in Figure 2.5.

There are several possible choices of probabilistic models that can be used in the BO framework. Some of these include Bayesian Neural Network (BNN) [23], random forest regression [24] and Parzen estimators [25]. However, these methods have relatively poor capability of predicting the posterior, especially in the scarce data regime that is sought after in electronic design optimization problems [26]. GPs, on the other hand, have many theoretical and practical advantages that align with the purposes of BO. From the theoretical point of view, it has been shown that Bayesian neural networks with one hidden layer converges to a GP when the number of hidden units tend to infinity [27]. Compared to BNNs, GPs are non-parametric models and have much smaller number of trainable hyperparameters. The non-parametric Bayesian nature of GPs also automatically implement Occam's Razor [28] and limit the complexity of the model. When this is combined with small number of learnable hyperparameters, GPs tend to avoid overfitting to scarce data, which makes them a very promising choice for the purposes of BO. Further, GPs with certain type of kernels have the universality property, meaning they can model any function over a compact set when the number of training data tends to infinity [29, 30]. Hence, we will use GPs as our probabilistic model in the BO framework.

After setting the model to a GP, we need the sampling strategy to lead the active learning

framework to find the global optima. In general, we want to sample in the promising regions of the sample space that we already know provides a high objective function value. This is known as *exploitation* of the best observed points, i.e. to select new samples that have high posterior mean. However, we also want to sample in the regions that we have not sampled yet to avoid missing other promising regions. This is known as *exploration* of the sample space, i.e. to sample points that have high posterior variance. The overall strategy we seek is the one that balances exploration and exploitation to achieve rapid convergence while covering the sample space very well.

One popular strategy in BO is called probability of improvement (PI) [31]. Here, the idea is to construct an *acquisition function* that uses posterior mean and variance of GPs to calculate probability of any point in the sample space being better than the best observed value. Hence, the argmax of this function gives the most probable point to improve over the current maximum point. PI can be written in closed form as

$$\begin{aligned} u_{\text{PI}}(x) =& p\left(f(x) \geq \widetilde{f}^*\right) \\ =& \Phi\left(\frac{\mu(x) - \widetilde{f}^*}{\sigma(x)}\right) \end{aligned} \tag{2.18}$$

where f^* is the maximum observed value, $\Phi(\cdot)$ is the cumulative distribution function (CDF) of standard normal, $\mu(\cdot)$ and $\sigma(\cdot)$ are the posterior mean and variance of GP as in Equation 2.13 and Equation 2.14, respectively. The optimization of this acquisition function to get the next sampling point is called as *auxiliary optimization*. Here, any optimizer can be used as its gradient can be calculated in closed-form and it is very cheap to evaluate.

Although the PI strategy gives the most probable point, it does not take into account how much this improvement is. When we consider not only the probability but also the value of improvement, we arrive at expected improvement (EI) strategy [32]. This results in an acquisition function that can be written as the expectation over the improvement over

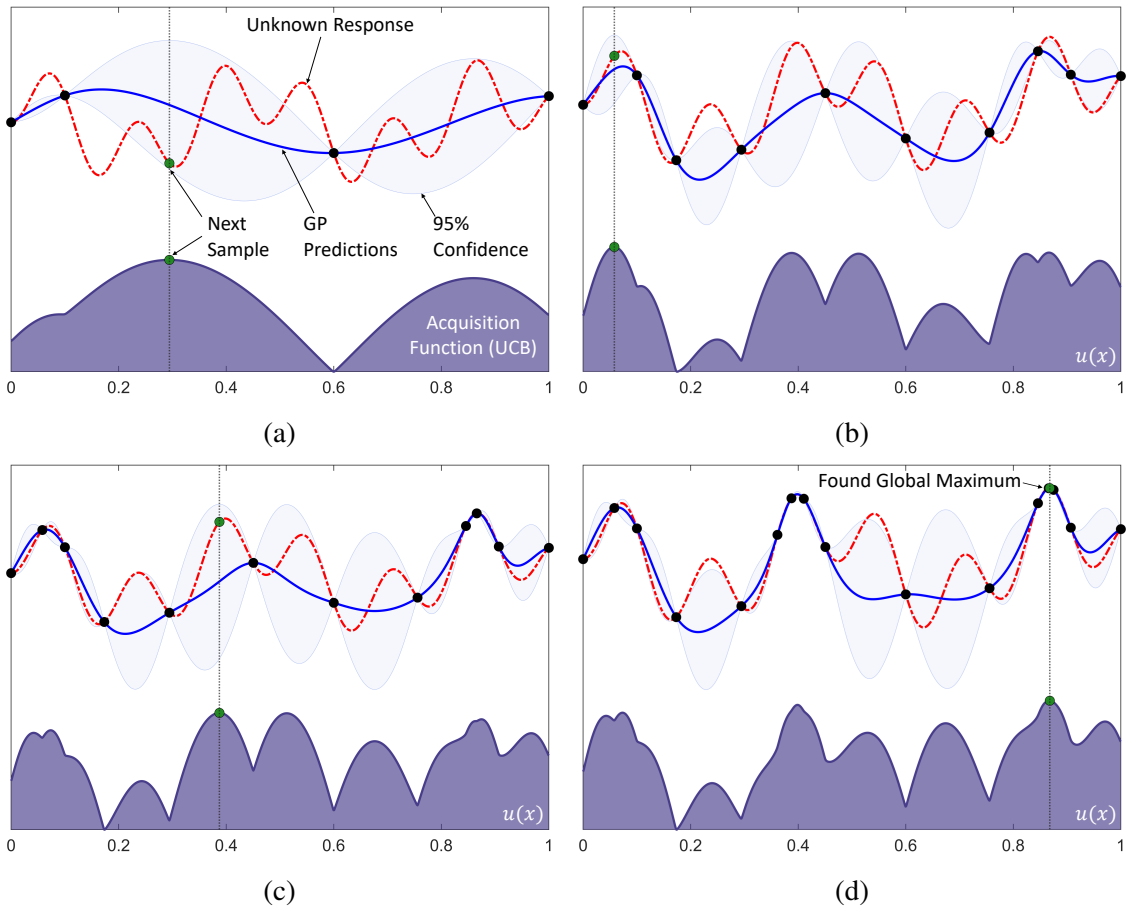


Figure 2.6: BO flow for maximizing a 1D toy function using UCB strategy. (a) t = 5. (b) t = 10. (c) t = 15. (d) t = 20

the current best point, given as

$$\begin{aligned} u_{\text{EI}}(x) =& \mathbb{E}\left[f(x) \geq \widetilde{f}^*\right]_{(f\sim GP)} \\ =& \left(\mu(x) - \widetilde{f}^*\right) \Phi\left(Z\right) + \sigma(x)\phi\left(Z\right) \end{aligned} \tag{2.19}$$

where $Z = (\mu(x) - \widetilde{f}^*)/\sigma(x)$ and $\phi(\cdot)$ is the probability distribution (PDF) of standard normal variable.

A more recently developed strategy is called upper confidence bound (UCB) criteria [33]. UCB strategy takes an alternative approach and pose the BO problem as a multi-armed bandit. The goal here is not to the find absolute best value, but find the best possible

value under a limited budget of function evaluations. This setting is of utmost importance for BO applied to packaging problems and we will elaborate on it in the following subsection. The UCB criteria can also be written in closed form as

$$u_{\text{UCB}} = \mu(x) + K\sigma(x), \quad K = \sqrt{2ln(2\pi M^2/(12\eta))} \tag{2.20}$$

where $(1 - \eta)$ is the probability of zero regret for UCB [33], i.e. probability of finding the absolute global optima. The flow of BO with UCB for maximizing a 1D non-convex (unknown) objective function is given in Figure 2.6. It can be seen that the sampling only occurs in the promising regions of the objective function, and as desired, the final GP model after 20 iterations of BO is only accurate in these promising regions.

2.6 Summary

In this chapter, we have provided a brief background on the concept of data-driven modeling and optimization for electronic design. We first presented the underlying idea behind building a deterministic data-driven model and covered how it can be achieved using neural networks. We then compared the formulation of the deterministic model-building setting to the principles of Bayesian learning and highlighted their differences in the context of non-linear regression. Finally, we have introduced Gaussian Process models as a way to derive Bayesian surrogate models and how it can be used to build a global optimization framework, namely Bayesian optimization.

CHAPTER 3

TRANSPOSED CONVOLUTIONAL NEURAL NETWORKS FOR PREDICTING FREQUENCY RESPONSES

Design of high-frequency systems often require handling multiple trade-offs to meet performance metrics. Therein arises the necessity to perform a thorough design space exploration (DSE) to evaluate various different circuit architectures, materials and geometries, where each design choice goes through a rigorous electrical characterization process. This step commonly includes characterizing the frequency response of passive components such as filters, connectors and interconnects through multi-scale, broadband 3D EM simulations, then using the resulting frequency response in further analysis.

Although essential for high-frequency design characterization, the involvement of 3D EM simulations in the DSE loop significantly increases the overall computational complexity, especially when the number of design choices and parameters increase. In order to create a learning-based model that maps such design parameters to the corresponding frequency response, the desired system response at each discrete frequency point need to be considered as a separate output dimension or a separate training sample. This high output dimensionality and/or high-number of data points creates a computational challenge for non-parametric Bayesian models such as GPs since the complexity of inverting the covariance matrix that is required to perform maximum likelihood estimation is $\mathcal{O}(M^3N^3)$ for a problem that has M output dimensions and N data points.

Parametric models, such as neural networks (NN), can better handle the increasing number of training data points and/or output dimensionality since the training data itself is not used for inference. Hence, once the training is done, the NN can be used as a standalone module to generate frequency responses in a large bandwidth with small frequency step sizes. However, existing NN based approaches have some limitations that needs to be

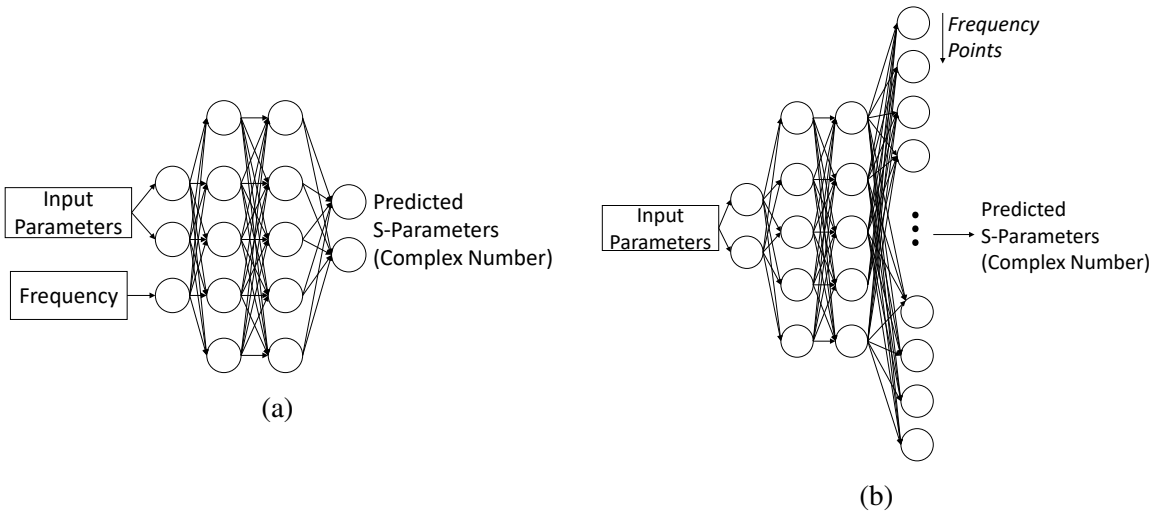


Figure 3.1: FCNN using (a) Frequency as input and (b) Frequency as Output

addressed before they can be used in a DSE loop to predict frequency responses.

To elaborate on this, consider two formulations that can be used to generate frequency responses using conventional fully-connected NNs (FCNN), i) treating frequency as a regular input parameter as in Figure 3.1a and ii) considering each frequency point as a separate output dimension as in Figure 3.1b.

In the former, different design settings are first evaluated at a given frequency band to create the training data. Each setting is then replicated, where each copy is associated to a particular frequency point. When the number of frequency points increase as in signal and power integrity (SI & PI) analysis of high-bandwidth systems, such replication results in a very large, but redundant training dataset that requires impractical memory to store and process during the training stage. As an example, consider the response of a 4-port structure at 2000 frequency points, parameterized with respect to 10 variables. To train a NN, we start by creating data, where we compute, say, 500 variable combinations and store their corresponding responses. For the frequency as input configuration in Figure 3.1a, we need to replicate each variable combination 2000 times to generate an input-output dataset, resulting in 1M data points. This creates a significant memory problem for training algorithms due to large data size. This formulation is therefore only applicable to narrow-

band responses, e.g. RF applications [34], or to smooth responses where the frequency band can be sparsely discretized without losing accuracy [35].

Considering each frequency point as a separate output parameter approach as in Figure 3.1b addresses the redundant parameter replication step and removes the memory problem. However, the large number of frequency points now creates a potentially bigger problem as it results in a very high-dimensional output space. For an N-port reciprocal structure evaluated at M frequency points, the total number of dimensions (outputs) are $D = 2MN(N+1)/2$, where the factor of 2 comes from real and imaginary parts of a complex number. Consider a very simple FFNN that has only 1 hidden layer with 10 neurons. The number of weights that connect this hidden layer to D outputs is then $10D$. For a 4-port structure at 2000 frequency points, D becomes 40K and number of weights for just a single hidden layer is 400K. Although the memory problem is reduced, learning that many weights can be challenging and can lead to overfitting. Overfitting is caused when random fluctuations in the training data are learned as being part of the model, which leads to erroneous predictions. A familiar example to overfitting is using a very high-order polynomial when the data can be better described by a lower-order polynomial.

To address the overfitting problem without losing predictive accuracy, we need to reduce the number of learnable parameters without giving up the model's representation capability. This can be done by exploiting the structure of the data. Although the frequency as output formulation becomes a very high-dimensional problem, it is highly structured in the sense that there exists a spatial correlation in the frequency axis, i.e. neighboring frequency points are highly correlated with each other. This spatial correlation can be highly exploited using convolutional neural networks (CNN).

In this chapter, we therefore propose a new NN architecture, named as Spectral Transposed Convolutional Neural Network (S-TCNN), to learn the frequency response of electronic devices by utilizing 1D convolution operations. Here, we exploit the spatial correlation in the frequency axis using convolution operations to learn the whole frequency

response with a reduced number of learnable parameters. The proposed S-TCNN model results in a model with reduced training complexity that better generalizes to the cases outside of the training data as compared to the FCNN counterparts in Figure 3.1. Further, we present a new loss function to replace the commonly used mean-squared error (MSE) for learning the frequency response. Here, the loss function we propose aims to minimize the loss between the predicted and the actual frequency response as a whole rather than the average loss at individual frequency points as in the case of MSE loss. We then show how S-TCNN architecture can be extended to include confidence intervals in its predictions by utilizing Bayesian learning techniques. To demonstrate the effectiveness of the proposed method, the presented S-TCNN model is used to accurately learn the frequency response of a solenoidal inductor with magnetic core and radar cross-section (RCS) of an aircraft.

3.1 Spectral Transposed Neural Network

3.1.1 Convolutional Layers

A convolutional layer in NNs aims to learn local patterns in the axes that contain a spatial correlation. In the context of frequency responses, this corresponds to searching for patterns such as resonances, ripples and flat regions in smaller frequency bands. CNNs achieve this by stacking multiple convolutional layers. This is achieved by employing a sliding inner product operation on small frequency bands and sharing the learnable weights in the hidden units across the whole frequency spectrum, which results in a reduced number of weights that need to be optimized during the training. Most commonly used form of CNNs utilize a 2D sliding product to consider spatial correlation in two axis of the data, such as the width and height of an image. For the cases of frequency responses, the spatial correlation to be exploited is along the frequency axis. Hence, a 1D operation needs to be utilized.

Unlike fully-connected layers where each neuron has a single weight, the neurons in convolutional layers form 1D arrays and now called kernels. Each kernel contains the learnable weights of the NNs and perform 1D sliding inner products in the frequency axis

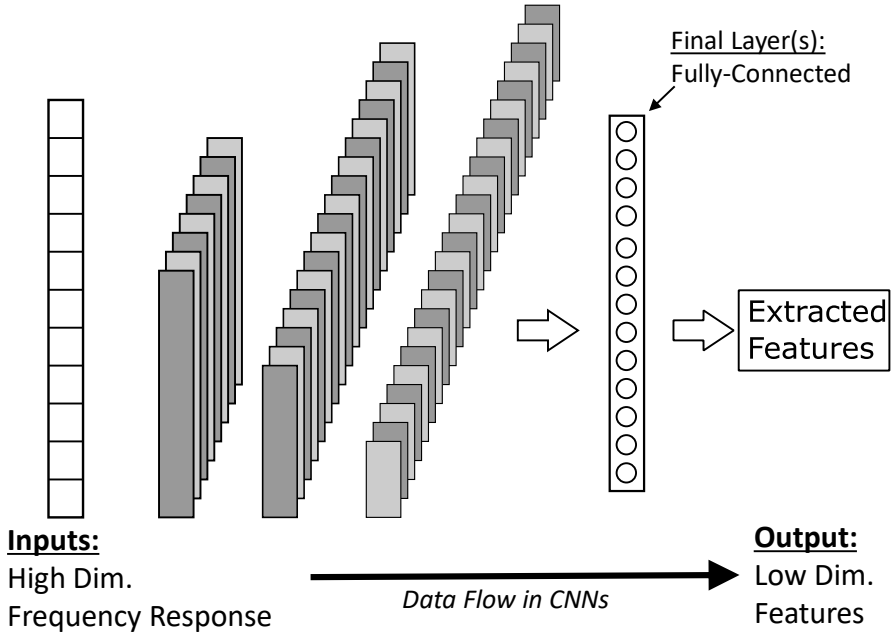


Figure 3.2: Conventional CNN architecture to downsample high-dimensional frequency responses into low-dimensional features.

followed by the non-linear activation function. The operation done by a single kernel is best understood by writing out the sliding product as a matrix multiplication as

$$\mathbf{y} = f(\mathbf{h} * \mathbf{x}) = f(\mathbf{H}\mathbf{x}) \tag{3.1}$$

with

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{H} = \begin{bmatrix} w_1 & w_2 & \cdots & w_k & 0 & \cdots & 0 \\ 0 & w_1 & w_2 & \cdots & w_k & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & w_1 & w_2 & \cdots & w_k \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} \tag{3.2}$$

where $(*)$ denotes the convolution operation, $f(\cdot)$ is the non-linear activation function, $\mathbf{H}$ is the convolution matrix of size $n \times m$ and $\mathbf{y}$ is the downsampled output of size $n = m - k + 1$. When the convolution operation is written as $f(\mathbf{H}\mathbf{x})$, each row of H represents the frequency axis. As the learnable weights $(w_{1,\ldots,k})$ in each row are the slid version of the same values, the weights are shared across the frequency axis. Just like FCNNs

use increased number of neurons to learn different features, the number of kernels can be increased in CNNs to learn different patterns.

3.1.2 Transposed Convolutional Layers

Such convolutional layers can be very useful for extracting features from the frequency response data since the data flow in regular CNNs is from the high-dimensional frequency response to low-dimensional features as in Figure 3.2. We want exactly the opposite of this flow to predict frequency responses, i.e. mapping from lower dimensional design parameters to high-dimensional frequency responses. This can be done by using the so-called *transposed convolutional layers*. Such layers perform learnable upsampling operations that preserves the spatial correlation in its output by considering a particular input as the result of convolution between the output and the learnable kernel [36]. Using the same matrix multiplication notation as in Equation 3.2, this can be written as

$$\mathbf{y} = f(\mathbf{h} *^{\intercal} \mathbf{x}) = f(\mathbf{H}^{\intercal}\mathbf{x}) \tag{3.3}$$

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \mathbf{H}^{\intercal} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ w_2 & w_1 & \ddots & \vdots \\ \vdots & w_2 & \ddots & 0 \\ w_k & \vdots & \ddots & w_1 \\ 0 & w_k & \ddots & w_2 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & w_k \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{3.4}$$

where $*^{\intercal}$ is the transposed convolution operation and $\mathbf{y}$ is the upsampled output of size $m = n + k - 1$. Note that the upsampling ratio can be increased by making use of strided transposed convolutions, where zeros are padded between input points along the convolution axis [36]. This allows to achieve higher upsampling ratios with less number of layers

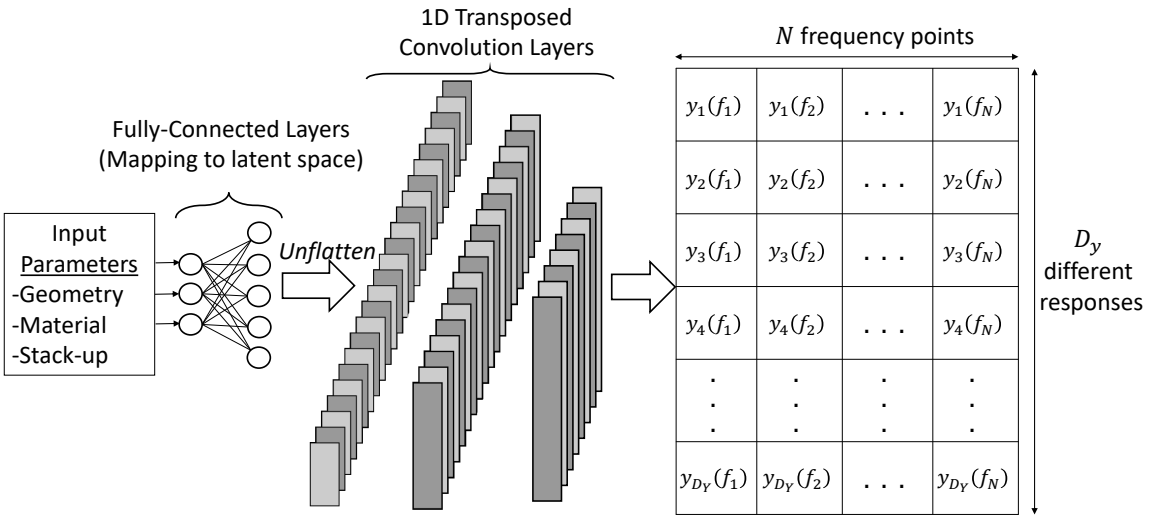


Figure 3.3: Proposed S-TCNN architecture to predict high-dimensional frequency responses from low-dimensional design parameters.

and helps control the depth the network.

One question that can be asked here is: How do we know an arbitrary design parameter can be upsampled to obtain corresponding frequency responses? The design parameters themselves might not be the best features to start the upsampling sequence. To make sure such operations would be successful, we first convert the actual inputs to a new domain that best describe the start of the sequence. Since it is not possible to know such transformation beforehand, we make use of fully-connected layers as the very first few layers in our network. The resulting architecture becomes as in Figure 3.3 and is called *Spectral Transposed Convolutional Networks* (S-TCNN), which takes design parameters as input, converts them to a latent space and then upsamples it using 1D transposed convolutions to obtain frequency responses.

3.1.3 Loss Function to Learn Frequency Responses

Once the architecture is finalized, we need to determine the loss function to be minimized to learn the model parameters. The most conventional loss function for training a NN is the

mean squared error (MSE), given as

$$L = \frac{1}{NK} \sum_{n=1}^{N} \sum_{k=1}^{K} (y_{n,k} - \hat{y}_{n,k})^2 \tag{3.5}$$

where $\hat{y}_{n,k}$ is the k^{th} dimension of the output of the network at n^{th} training sample. For frequency responses, k represents the discrete frequency points. Observe that in Equation 3.5, if we change the ordering of data and frequency axes in $\hat{y}_{n,k}$ to get $\hat{y}_{k,n}$, the MSE loss will be exactly the same. The MSE function then can not distinguish between different axes, which corresponds to network being trained to predict individual frequency points, rather a frequency response as intended.

To address this problem, we propose a modified loss function to change the behavior of the S-TCNN model to reconstruct the frequency response as a whole, which can be written as

$$L_{freq} = \frac{1}{N} \sum_{n=1}^{N} \sqrt{\frac{1}{K} \sum_{k=1}^{K} (y_{n,k} - \hat{y}_{n,k})^2}. \tag{3.6}$$

Formally, the loss function in Equation 3.6 is the scaled ℓ^2-norm of the error between the predicted and the actual frequency response, averaged over N different frequency responses in the training set. This ensures that the S-TCNN model learns the mapping from a certain input vector to the frequency response as a whole, rather than the mapping to k different frequency points. As we show in the subsequent section, L_{freq} in Equation 3.6 improves the accuracy of the network not just for the S-TCNN model, but also for the regular FCNNs for the task of predicting frequency responses.

3.2 Application 1: Modeling Embedded Solenoidal Inductors

As an application example, we consider learning to generate frequency response of a solenoid inductor with magnetic core that is integrated on the top metal layer of a silicon interposer based 2.5D heterogeneously integrated system as in Figure 3.4(a). The electri-

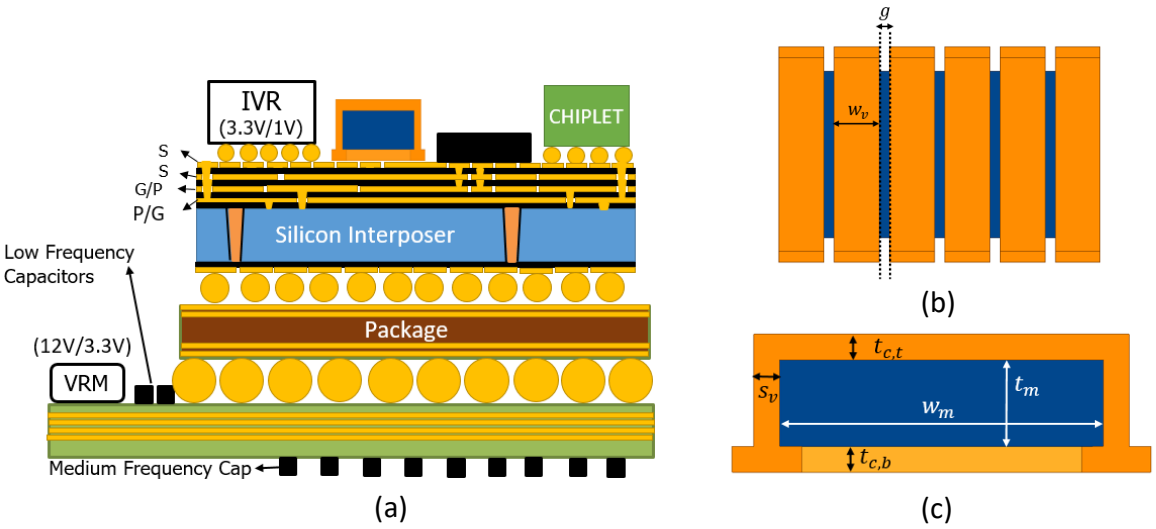


Figure 3.4: The power delivery architecture under consideration. (a) Stack-up of the overall system. (b) Top view of the solenoid inductor. (c) Side view of the inductor.

Table 3.1: Control Parameters of the Inductor in Figure 3.4

Parameter		**Unit**	**Min**	**Max**
Gap between windings	g	mil	2	20
Number of windings	N		3	13
Size of via	s_v	μm	50	103
Copper Trace Width	w_c	mil	2	20
Copper Thickness Bottom	$t_{c,b}$	μm	35	170
Copper Thickness Top	$t_{c,t}$	μm	35	170
Magnetic Core Thickness	t_d	μm	50	650
Magnetic Core Width	w_d	μm	50	350

cal characteristics of such inductors have a significant impact on systems equipped with integrated voltage regulators (IVR). We therefore derive a model to predict its inductance and effective series resistance (R) such that it can be later combined with circuit models of DC-DC converts to perform rapid DSE for power delivery performance. We parameterize the geometry of the inductor by 8 parameters as shown in Figure 3.4(b) and Figure 3.4(c) and corresponding bounds for each parameter are given in Table 3.1.

To create the training data, we determine 1000 samples based on Latin Hypercube Sampling (LHS) and feed into a 3D EM solver, Ansys HFSS, to extract the frequency depen-

dent inductance and resistance at 200 frequency points between 10 MHz and 500 MHz. We then use 800 of the samples for training and the remaining 200 for validating the model. We compare the performance of S-TCNN model with FCNN that uses frequency as output formulation in Figure 3.1b and train both models using both MSE loss in Equation 3.5 and the proposed L_{freq} loss in Equation 3.6. To assess the quality of models, we use normalized mean squared error (NMSE) averaged over each frequency response in the validation set, which can be written as

$$NMSE = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{\sum_{k=1}^{K}\left(y_{n,k}-y_{n,k}\right)^2}{\sum_{k=1}^{K}\left(y_{n,k}-\frac{1}{K}\sum_{k=1}^{K}y_{n,k}\right)^2}\right) \tag{3.7}$$

where N is the size of the validation set and k is the frequency axis. We choose NMSE metric to evaluate the models as it is highly intuitive and provides a normalized scale for different outputs, i.e. $L(f)$ & $R(f)$. Here, an NMSE value of 1.0 means that the model is able to predict no better than the mean of each frequency response.

Table 3.2 summarizes NMSE values for each model and training loss function. The proposed S-TCNN model results in 10.8% improvement in prediction accuracy as compared to the most commonly used architecture in the literature, FC-NN model that is trained on MSE loss. The use of the proposed L_{freq} loss function resulted in 3.2% and 5.1% improvement in prediction accuracy for FC-NN and S-TCNN models that are trained using the MSE loss, showing its efficacy for predicting frequency responses. This is further demonstrated in Figure 3.5. Here, it can be seen that in addition to the final value, the convergence of validation NMSE is faster and more robust for both FC-NN and S-TCNN when trained with L_{freq} loss. This shows that the proposed loss function reduces the training complexity and better represents the generalization capability of the network to unseen frequency responses as compared to the training with MSE loss. When both models are trained with the L_{freq} loss, S-TCNN showed 5.2% increased accuracy compared to FC-NN, which shows its ability to exploit the spatial correlation in the frequency axis to learn the patterns of $L(f)$

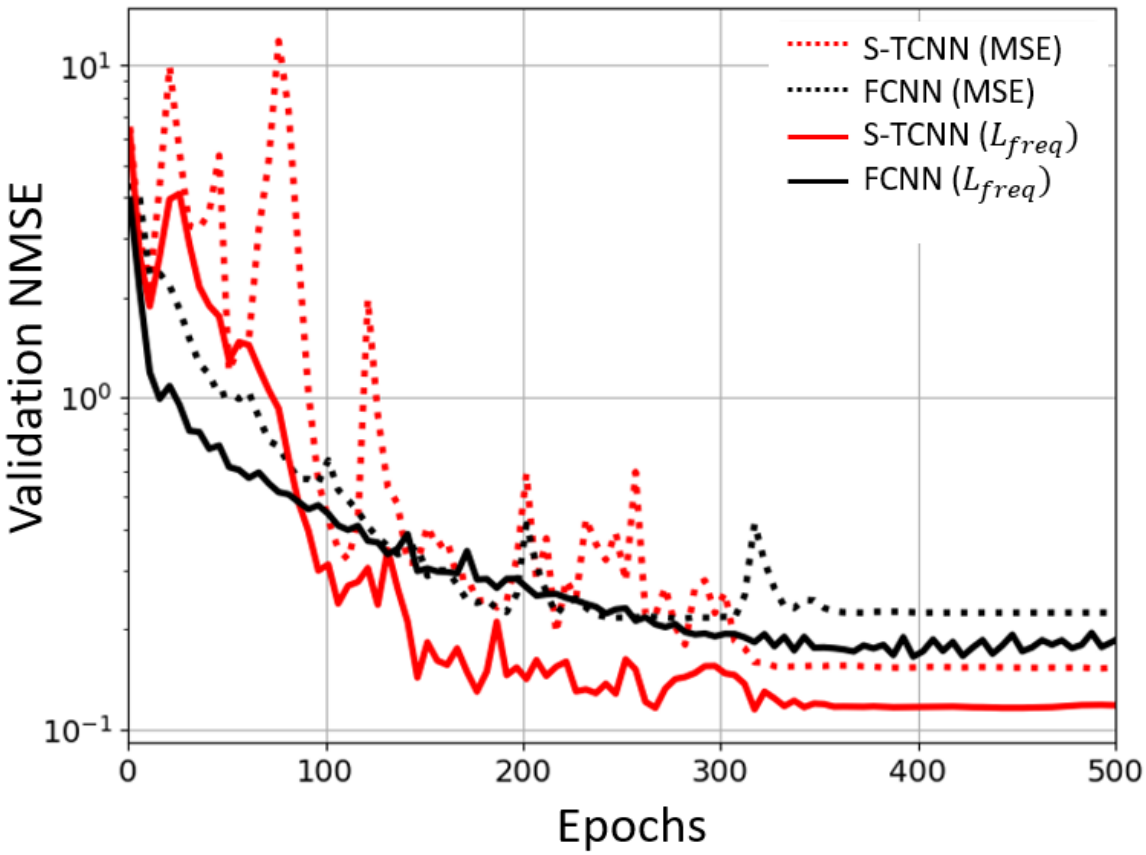


Figure 3.5: Convergence of validation loss for different models on IVR example.

Table 3.2: Performance comparison of different models and loss functions on IVR example.

	FCNN		**Proposed S-TCNN**	
	MSE	L_{freq}	**MSE**	L_{freq}
Validation NMSE	0.228	0.177	0.152	0.120
Run Time *(for 1k freq. responses)*	0.01 sec		1.503 sec *(HFSS: $\sim$ 25 hours)*	

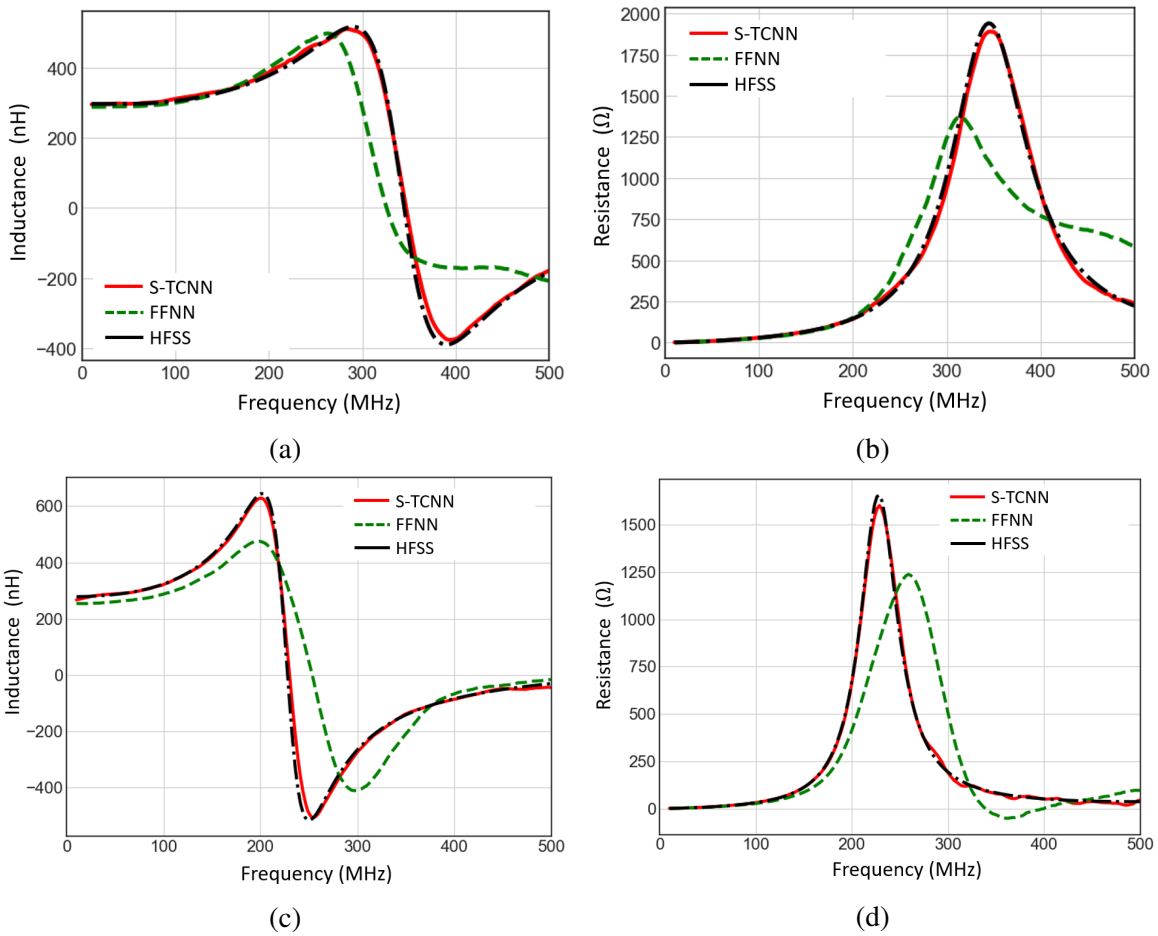


Figure 3.6: Comparison $L(f)$ and $R(f)$ obtained by S-TCNN and FCNN to EM simulations for two test cases. (a), (b) Test case #1. (c), (d) Test case #2.

& $R(f)$. As further illustrated in Figure 3.6, the predictions done by S-TCNN capture the behavior of self-resonance patterns seen in $L(f)$ & $R(f)$ very accurately, whereas FCNN shows poor performance for learning this pattern. In terms of the run times, S-TCNN took 1.503 sec to generate frequency responses of 1k different designs as compared to 25 hours by HFSS. FCNN was slightly faster as it took 0.01 sec for the same task, however, the difference is insignificant for the purposes of optimization.

3.3 Variational Bayes Approximation to Obtain Confidence Intervals

Although convolutional type neural networks can be very useful for predicting frequency responses, the model is deterministic and it is not possible to asses the quality of the pre-

dictions at inference time. This limits the use of such models in practice as design choices made through these predictions can be misleading and result in low performance or even system failure. Hence, a feedback from the model indicating the quality of the predictions in the form of confidence bounds should be established.

There are several ways to convert NNs, including S-TCNN, into a Bayesian model where such confidence intervals can be established. These techniques are broadly called as Bayesian Neural Networks (BNN), and are based on converting deterministic parameters of the NN, i.e. weights and biases, to a distribution at each or only specific layers of the overall network architecture. Here, we demonstrate the BNN formulation on a conventional fully-connected layer, but it can be extended to any NN layer that has learnable parameters.

Let $\mathbf{W}$ and $\mathbf{b}$ denote the weights and biases of a fully-connected NN layer. For an input vector $\mathbf{X}$, the output of this layer can be written as

$$\mathbf{y} = f(\mathbf{W}\mathbf{X} + \mathbf{b}) \tag{3.8}$$

where $f(\cdot)$ is the non-linear activation function such as $\tanh(\cdot)$. In conventional NNs, the learnable parameters in Equation 3.8, $\mathbf{W}$ and $\mathbf{b}$, are deterministic numbers, meaning that the output $\mathbf{y}$ is also a deterministic number.

In BNNs, we consider the learnable parameters (collectively referred as $\theta = \{\mathbf{W}, \mathbf{b}\}$) as samples from a distribution. Instead of making a single prediction with a fixed θ, we make multiple predictions where each prediction uses a different sample of θ to form the output. More formally, a BNN is a NN with a prior distribution on its weights ($p(\theta)$) [27]. Given a dataset $\{\mathbf{X}, \mathbf{y}\}$, the goal of training a BNN is then to learn the posterior distribution over these network parameters $p(\theta \mid \mathbf{X}, \mathbf{y})$. The posterior distribution over the predicted output, $p(y^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y})$, given a new input vector, $\mathbf{x}^*$, is then calculated as

$$p(y^* \mid \mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int p(y^* \mid x^*, \mathbf{X}, \mathbf{Y}, \theta) p(\theta \mid \mathbf{X}, \mathbf{Y}) d\theta. \tag{3.9}$$

The challenge in BNNs is that the parameter posterior, $p(\theta \mid \mathbf{X}, \mathbf{y})$, is analytically intractable, which prevents exact computation of the predictive posterior. Hence, we approximate the posterior in Equation 3.9 using an approximate Bayesian learning technique called as *variational inference*. Here, the goal is to learn a fast-to-compute approximate distribution, $q(\theta \mid \mathbf{X}, \mathbf{Y})$, that is as close as possible to the intractable parameter posterior. This gives a loss function to use while training the BNN, that is, minimizing the Kullback-Leibler (KL) divergence between the two distributions, written as

$$\mathcal{L} = \text{KL}(q(\theta \mid \mathbf{X}, \mathbf{Y}) \mid p(\theta \mid \mathbf{X}, \mathbf{y})) \tag{3.10}$$

After some algebraic manipulation using Jensen's inequality [37], the loss function can be re-written as

$$\mathcal{L} = \underbrace{\int q(\theta \mid \mathbf{X}, \mathbf{Y}) p(\mathbf{y} \mid \mathbf{x}, \theta)}_{\text{Approximated using MC integration}} - \underbrace{\text{KL}(q(\theta \mid \mathbf{X}, \mathbf{Y}) \mid p(\theta))}_{\text{Closed-form expression based on prior}} \tag{3.11}$$

where $p(\theta)$ is the prior distribution over NN parameters. Note that the first term in Equation 3.11 is approximated using Monte Carlo (MC) integration, i.e. sampling from $q(\theta|\mathbf{X}, \mathbf{Y})$ N-times and taking its expectation. The second term is KL divergence between the prior over network parameters and its approximation. Since we choose a simple and fast-to-compute distribution for $q(\theta \mid \mathbf{X}, \mathbf{Y})$, without loss of generality, the KL divergence term can be computed in a closed-form. After the training is finalized, the inference equation in Equation 3.8 can be replaced with

$$q(y^* \mid x^*, \mathbf{X}, \mathbf{Y}) = \int p(y^* \mid x^*, \theta) q(\theta \mid \mathbf{X}, \mathbf{Y}) d\theta. \tag{3.12}$$

The formulation presented so far is in terms of generic distributions. In practice, we need to choose the type of $q(\cdot)$ to train our model and make inference. In this chapter, we follow [38] and choose a Bernoulli distribution for $q(\cdot)$. This enables us to implement the

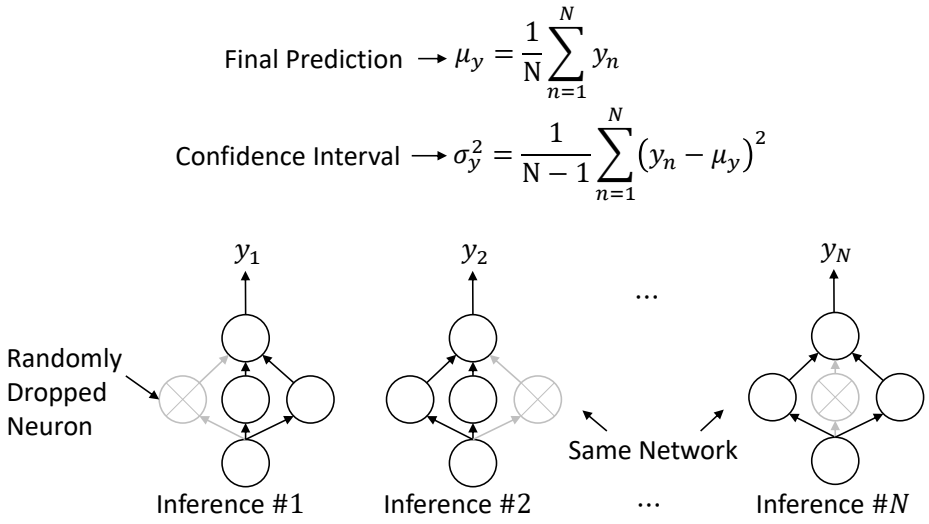


Figure 3.7: Dropout at inference time to obtain confidence intervals around NN predictions.

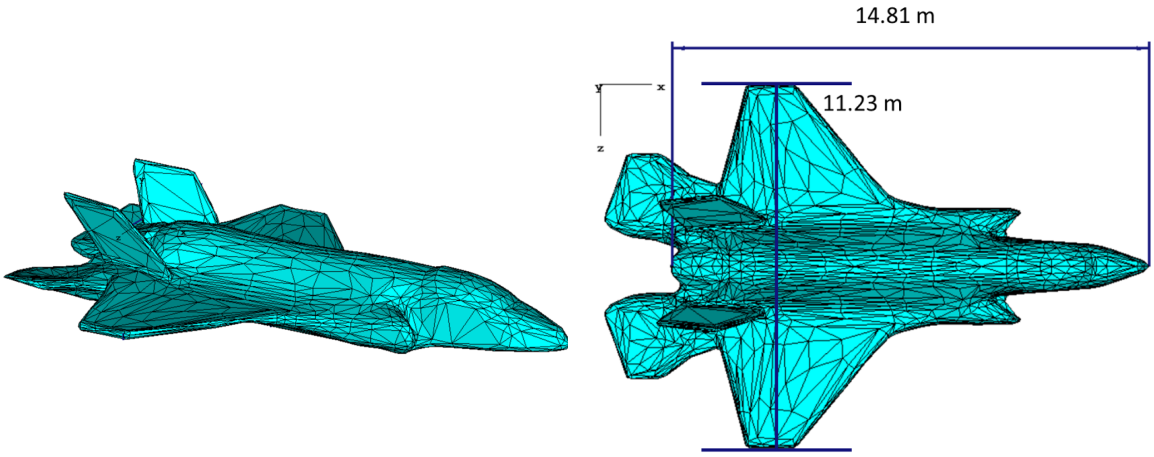
formulation given in Equation 3.9 through Equation 3.12 in a fairly simple manner using the following steps:

1. Determine the NN architecture (FCNN, STCNN, etc).

2. Train the NN using as in the deterministic case, but using dropout [39] after each layer in the network architecture.

3. At inference time, instead of disabling dropout and performing a single forward-pass, keep randomly dropping network parameters as in training phase and perform N forward-passes. Since we drop a different parameter at each forward pass, the predicted output will be different at each pass. The mean of N different passes, μ_y, is then the final predicted value and their variance, σ_y^2, is the uncertainty around the prediction as in Figure 3.7.

Since the dropout technique can be coupled with any NN architecture, we use it in conjunction with the S-TCNN architecture and call it as Bayesian S-TCNN. The resulting Bayesian S-TCNN model can then be used to predict frequency responses along with confidence bounds around the predictions to account for uncertainty in the model parameters.



(a) (b)

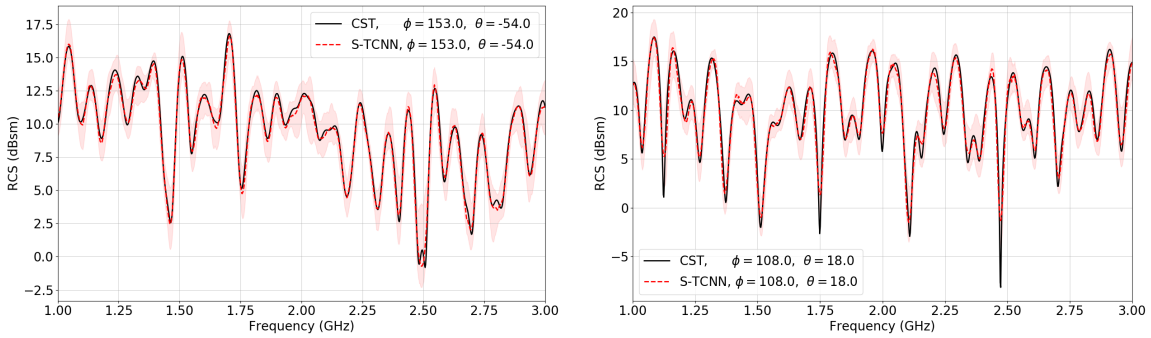
Figure 3.8: 3D meshed view of the aircraft. (a) Isotropic view. (b) Top View (Model courtesy: Eric Huang).

3.4 Application 2: Modeling Radar Cross-Section of Aircrafts

As an application example to demonstrate effectiveness of the Bayesian S-TCNN model, we consider parameterizing the frequency-dependent radar cross-section (RCS) of an aircraft as in Figure 3.8. RCS is electromagnetic signature of objects and used in radar systems to detect an approaching aircraft for a wide range of angles. Due to the large size of the aircraft and small meshing required to capture its multi-scale structure, computing its frequency-dependent RCS at different angles (θ and ϕ) requires a significant computational complexity. In this section, we therefore derive a Bayesian S-TCNN model to build a compact RCS model and predict RCS frequency response between 1-3 GHz given 2 input parameters, i.e. the angle of observation θ and ϕ.

In order to create the training data, we use CST to simulate 440 RCS frequency responses by sweeping $-180^\circ < \phi < 180^\circ$ and $-90^\circ < \phi < 90^\circ$ at 10° increments. We then train two seperate models, S-TCNN and Bayesian S-TCNN, to predict the RCS data at an arbitrary angle of observation and compare their performances. Note that the network architectures for both models are taken as the same for a fair comparison.

As can be seen in Figure 3.9, the Bayesian S-TCNN model is able to capture the highly



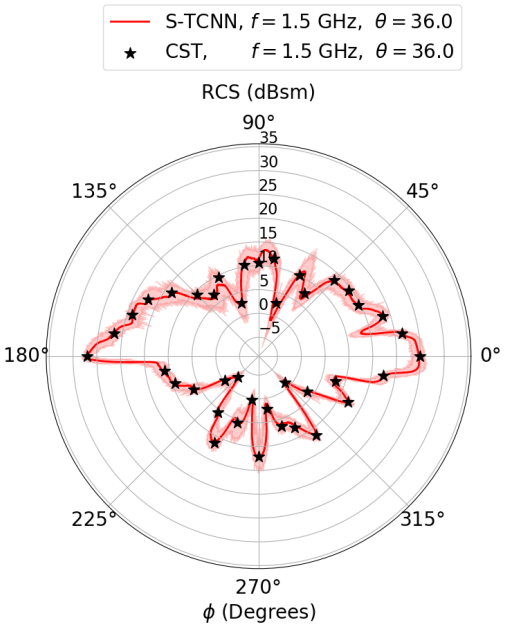
(a) (b)

Figure 3.9: Comparison of Bayesian S-TCNN and CST to predict RCS frequency-response. Shaded areas represent 95% confidence bounds.

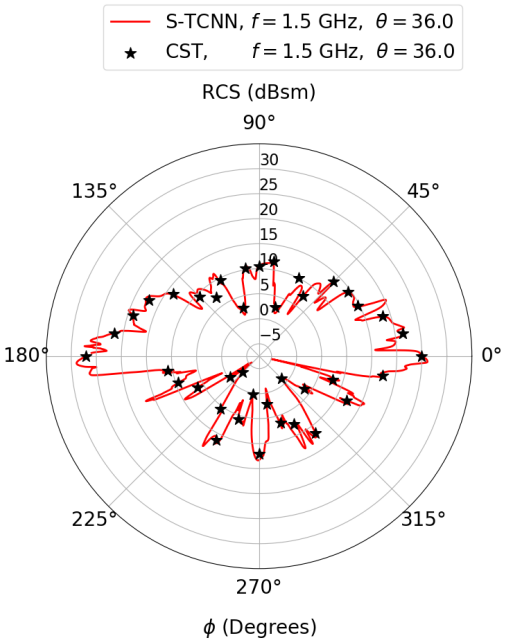
non-linear RCS frequency response at multiple angles. The NMSE for the two models are calculated to be 3.1% and 4.1% for Bayesian S-TCNN and S-TCNN, respectively, indicating good predictive accuracy can be achieved using both models. However, the confidence bounds obtained using the Bayesian S-TCNN provides a highly-valuable feedback in the sense that more training data are required to reduce model uncertainty. This can be observed from Figure 3.10. Although the predictive accuracy of the S-TCNN model is high, its predictions at 2.6 GHz is observed to be less accurate compared to 1.5 GHz. Since it is a deterministic model, there is no way to assess if there is any other frequency point or observation angle that might also provide erroneous predictions. The confidence bounds provided by the Bayesian S-TCNN model, on the other hand, indicates that model is more confident of its predictions at 1.5 GHz compared to 2.6 GHz, pointing to a need to collect more training data at 2.6 GHz to increase its prediction capability.

3.5 Conclusion

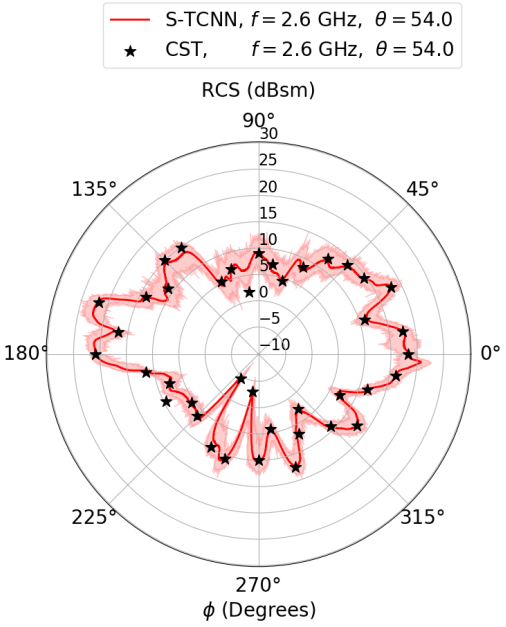
In this chapter, we have proposed a new method to learn the non-linear mapping from control parameters of an electronic device to its frequency response, named as Spectral Transposed Convolutional Neural Network (S-TCNN). Unlike previous methods in the literature, we've shown that the spatial correlation of the frequency spectrum can be exploited



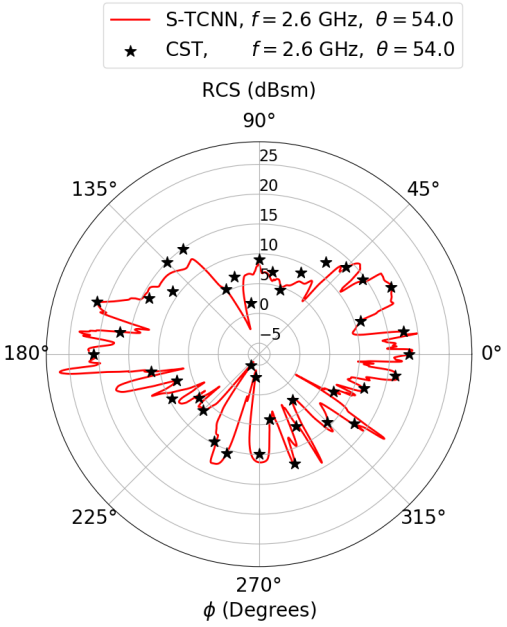
(a)



(b)



(c)



(d)

Figure 3.10: Comparison of Bayesian S-TCNN and S-TCNN to predict RCS at different observation angles. (a, c) Bayesian S-TCNN. (b, d) S-TCNN. Shaded areas represent 95% confidence bounds.

by using transposed convolutional layers to reduce number of learnable parameters in the model, thus, reducing the training complexity. Further, we have proposed a new loss function that is better suited to the reconstruction of frequency response as a whole rather than at individual points. For the application to solenoidal inductor with magnetic core, the proposed S-TCNN model showed 10.8% improvement in validation loss compared to the commonly used fully-connected networks, while the proposed loss function alone provided 5.1% improvement for the FC-NN model and 3.2% improvement for the S-TCNN model.

Further, we have demonstrated how to convert the S-TCNN architecture into a Bayesian model to obtain confidence intervals around the predicted frequency responses. When applied to modeling RCS frequency response of an aircraft, we have shown that although the predictive accuracy of both models were similar to each other, the predictions done by deterministic models can be misleading at inference time. Confidence bounds provided by the Bayesian S-TCNN model accurately captures the uncertainty of the model, hence, provides a quantitative measure to assess if the model predictions can be trusted or not in practice.

CHAPTER 4
PHYSICALLY CONSISTENT NEURAL NETWORKS FOR PARAMETERIZING S-PARAMETERS

In the previous chapter, we have presented a NN architecture to parameterize frequency responses such as scattering parameter (S-Parameter) matrices with respect to some design variables. We showed that the use of presented methodology can lead to rapid and accurate design space exploration (DSE) of electronic systems.

In many packaging problems, the DSE is performed in multiple domains in a two-step fashion, 1) performing frequency domain simulations using 3D EM solvers to generate frequency response of passive structures such as interconnects and 2) using the generated frequency response in the form of S-Parameters in a circuit simulator to be evaluated with the active components such as I/O drivers and voltage regulators in time-domain.

However, previous NN based modeling strategies, such as S-TCNN in Figure 3.3 and FCNN that uses frequency at input or output as in Figure 3.1, solely focus on numerically matching the predicted S-Parameters to the data source and overlook the underlying physical phenomena represented by the data. In the case of multi-port S-Parameters of passive microwave devices and networks, this can result in NN predicted S-Parameters to be non-causal and non-passive, which prevents the predictions to be used in subsequent time-domain and/or frequency-domain simulations for multi-domain DSE purposes. Hence, focus of NN based parameterization of S-Parameters should not solely be error reduction with respect to the data source, but also use the domain-knowledge we have to preserve the physical consistency, i.e. causality and passivity, of the predictions to enable an extended scope of use cases. Compared to the knowledge-based methods that make use of a coarse model to improve the overall prediction accuracy [40, 41], the knowledge in this chapter is used as a *constraint to be enforced on the predictions* based on the physical phenom-

ena that holds for S-Parameters of any passive microwave device and does not rely on a problem-specific coarse model.

In this chapter, we therefore *develop physically consistent NNs* to learn a physical representation between input parameters and broadband S-Parameters while minimizing the numerical error with respect to the training data. We propose two new layers to be used in a NN, namely *causality and passivity enforcement layers*. In the causality enforcement layer (CEL), we utilize Kramers-Kronig relations and use Hilbert transform to reconstruct the imaginary part of each element in the predicted S-Parameter matrix to ensure the time-domain impulse response matrix is causal. In the passivity enforcement layer (PEL), we enforce largest singular value of the predicted S-Parameter matrix at each frequency point to be less than 1 by using a minimum-phase passivity enforcement approach. We then demonstrate the proposed method on 3 different design applications that emerge in high-speed channel design.

The rest of this chapter is structured as follows: Section 4.1 provides background on S-Parameter causality and passivity, Section 4.2 presents the proposed NN architecture with CEL and PEL, Section 4.3 shows the application of the proposed model to a differential plated through hole (PTH) structure in package core, Section 4.4 presents the application to a differential stripline model in package, Section 4.5 presents the application to a ball-grid-array (BGA) model for package-to-board transition, followed by conclusion in Section 4.6.

4.1 Background on Causality and Passivity

Physical consistency of S-Parameters is a well-studied subject [42, 43, 44, 45]. In this section, we provide a brief summary of the fundamental concepts related to causality and passivity and introduce the notations we use in the subsequent sections.

4.1.1 Causality of S-Parameters

S-Parameter matrix of a P-port and linear time-invariant (LTI) system is said to be causal if every element in the time-domain impulse response matrix can not produce an output before the input signal, i.e.

$$h_{ij}(t) = 0, \;\; t < 0, \;\; \forall i, j \in P. \tag{4.1}$$

This condition is satisfied when even and odd parts of the transfer functions are related as

$$\begin{aligned} h_{ij}(t) &= h_{ij}^{(e)}(t) + h_{ij}^{(o)}(t) \\ &= h_{ij}^{(e)}(t) + sign(t) h_{ij}^{(e)}(t) \end{aligned} \tag{4.2}$$

where $h^{(e)}(t)$ and $h^{(o)}(t)$ are the even and odd parts of $h(t)$, respectively, and $sign(t)$ is the signum function that equals to 1 when $t > 0$ and -1 when $t < 0$. Taking the Fourier transform of Equation 4.2 gives

$$\begin{aligned} H_{ij}(f) &= \mathcal{F}\left(h_{ij}(t)\right) \\ &= H_{ij}^{(e)}(f) + \frac{1}{j\pi f} * H_{ij}^{(e)}(f) \end{aligned} \tag{4.3}$$

where $*$ is the convolution operation. As $H_{ij}^{(e)}(f)$ is the Fourier transform of an even function, it is real valued, leading to

$$\begin{aligned} \mathrm{Im}\left\{H_{ij}(f)\right\} &= \frac{-1}{\pi f} * \mathrm{Re}\left\{H_{ij}(f)\right\} \\ &= \frac{-1}{\pi} \int_{-\infty}^{\infty} \frac{\mathrm{Re}\left\{H_{ij}(\widetilde{f})\right\}}{f - \widetilde{f}} \mathrm{d}\widetilde{f} \end{aligned} \tag{4.4}$$

The convolution integral in Equation 4.4 is also known as the Hilbert transform. Rewriting Equation 4.4 for real and imaginary parts leads to the well-known Kramers-Kronig relations

[46], written as

$$V(f) = -\mathscr{H}\{U(f)\} = \frac{-1}{\pi}\int_{-\infty}^{\infty}\frac{U(\widetilde{f})}{f-\widetilde{f}}\,\mathrm{d}\widetilde{f}$$
$$U(f) = \mathscr{H}\{V(f)\} = \frac{1}{\pi}\int_{-\infty}^{\infty}\frac{V(\widetilde{f})}{f-\widetilde{f}}\,\mathrm{d}\widetilde{f} \tag{4.5}$$

where $H(f) = U(f) + jV(f)$ and $\mathscr{H}\{\cdot\}$ is the Hilbert transform operator. Hence, the causality condition in Equation 4.1 is satisfied if and only if the real and imaginary part of each element in the S-Parameter matrix satisfies Kramers-Kronig relations and is related through the Hilbert transform.

4.1.2 Passivity of S-Parameters

Although the causality condition is of utmost importance for using S-Parameter representation in any time-domain characterization, a significant portion of microwave analysis is performed solely in the frequency-domain. The passivity condition, stating that multi-port S-Parameters for a passive network can not generate energy, is of paramount importance for both time and frequency-domain characterization since any violation of passivity can lead to unstability in time-domain [47], and can be significantly amplified by active components and/or when cascaded to other non-passive S-Parameter blocks.

A P-port S-Parameter matrix defined within the frequency band Ω is said to be passive if and only if it is bounded as

$$S^H(f)S(f) \leq I, \;\; \forall f \in \Omega \tag{4.6}$$

where $(\cdot)^H$ is the Hermitian transpose operator. This condition can be conveniently checked by obtaining the singular values of $S(f)$ via singular value decomposition of the S-Parameter

matrix at every frequency point as

$$SVD\left[S(f)\right] = U(f)\Sigma(f)V^{-1}(f) \tag{4.7}$$

where

$$\Sigma(f) = \begin{bmatrix} \sigma_1(f) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_P(f) \end{bmatrix} \tag{4.8}$$

is the ordered singular value matrix with $\sigma_1(f) > \sigma_2(f) > \ldots > \sigma_P(f)$. The passivity condition in Equation 4.6 then can be re-written as

$$\sigma_1(f) \leq 1, \;\; \forall f \in \Omega. \tag{4.9}$$

Throughout this chapter, we will use Equation 4.9 to check the passivity of a given S-Parameter matrix.

4.2 Proposed Causal and Passive Neural Network Architecture for Parametrizing S-Parameters

In this section, we present the proposed neural network architecture that guarantees the predicted S-Parameters are physically consistent, i.e. satisfy the conditions in Equation 4.5 and Equation 4.9, while maximizing numerical accuracy with respect to the training data that are obtained from full-wave EM simulations.

A high-level block diagram of the proposed model is given in Figure 4.1 and consists of four blocks, namely a base network, learnable smoothing layer, causality enforcement layer (CEL) and passivity enforcement layer (PEL). The base network and the learnable smoothing layer contains all the learnable parameters in the overall model that are trained to minimize the training loss function, while CEL and PEL ensures physical consistency of the predictions. The output of the last block of the model (PEL) is compared to the training

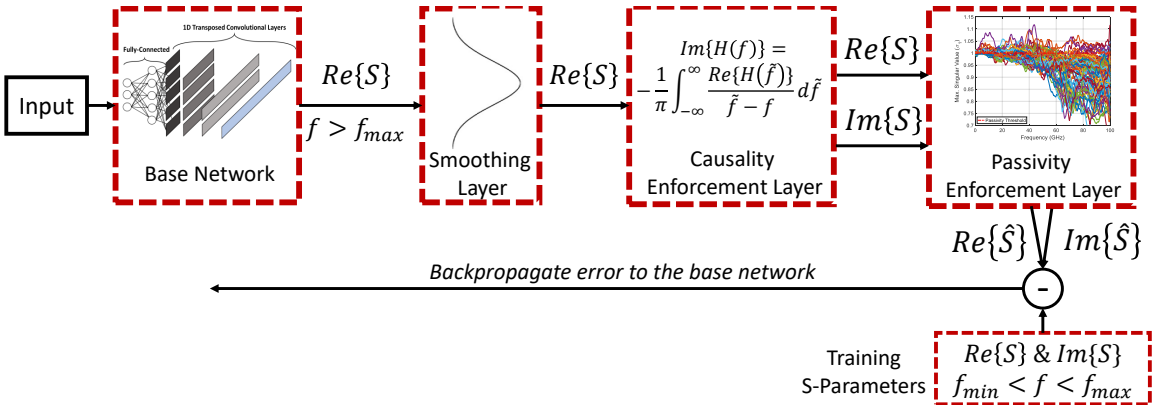


Figure 4.1: Proposed physically consistent neural network model for passive and causal parameterization of S-Parameters.

data and the error is backpropagated to update the learnable parameters. CEL and PEL are derived to ensure training and inference complexity of the NN is not substantially increased by not increasing the number of learnable parameters, vanishing or exploding gradients or increased computational complexity.

4.2.1 Base Network

The base network contains majority of the learnable parameters in the model. It uses the frequency as output formulation where each frequency point is considered to be a separate output dimension to be predicted. Hence, the base network is responsible for taking the input parameters of the model and upsampling it to output the multi-port frequency response.

Although any NN architecture that use frequency as output formulation can be used as the base netwrok, in order to handle dimensional frequency responses, we use the S-TCNN model given in Figure 3.3 and explained in detail in Chapter 3. As the output of the base network is to be passed into subsequent layers and not compared to the training data directly as in Figure 4.1, it becomes a latent variable. For reasons that will become clearer in the later subsections, we treat this latent variable to be the extrapolated version of the real part of the predicted S-Parameter matrix. The overall operation of the base network

can then be written as

$$\mathbf{y_{BN}} = f_{BN}(\mathbf{x}) \tag{4.10}$$

where $\mathbf{x}$ denotes the input parameters and $\mathbf{y_{BN}}$ is the output with size $N_d \times D_y \times (NM+1)$, where N_d is the number of data in a batch, D_y is the number of output channels, i.e. unique elements in S-Parameter matrix, N is the number of frequency points in the training data and M is the extrapolation factor. It should be noted that in general, we will be working with P-port reciprocal systems, hence, the number of channels will be equal to $D_y = 2(P(P+1)/2)$ unless additional symmetry exists, and the factor of 2 comes from real and imaginary decomposition. The output $\mathbf{y_{BN}}$ is forwarded to the learnable smoothing layer.

4.2.2 Learnable Smoothing Layer

Although strided convolution operation allows to increase upsampling ratio and reduce the number of layers, it can result in uneven overlaps where parts of the output vector are calculated using a larger portion of the input vector than others. This effect is recognized as the checkerboard artifacts [48]. Although the network can ideally learn and adjust its weights to cancel out the checkerboard artifacts, similar to [48], we have observed that it is not completely avoided when predicting the frequency responses.

In practice, one can smoothen the frequency response in the inference stage after the network forms its output. However, it is not possible to know which type of filter, and with what settings, should be used for this purpose. In this chapter, we therefore propose to use an adaptive Gaussian smoothing filter as part of the overall network, and assign the standard deviation of it as a learnable parameter of the model for which we learn during the training just as another weight or bias in the base network.

In particular, we use a seperate filter for each channel of the $\mathbf{y_{BN}}$. This can be written as

$$\mathbf{y_{SL}} = \mathbf{y_{BN}} \circledast \frac{1}{\sqrt{2\pi}\sigma_i} e^{-\frac{-l^2}{2\sigma_i^2}} \tag{4.11}$$

where

$$\boldsymbol{\sigma} = [\sigma_1, \sigma_2, ..., \sigma_i, ..., \sigma_{D_y}] \tag{4.12}$$

and l denotes the discretization grid to compute the Gaussian kernel and σ_i is the learnable standard deviation of the Gaussian kernel per channel. $\circledast$ denotes channel-wise convolution operation, i.e. separate filtering of each frequency response in channels of $\mathbf{y_{BN}}$. In neural network terminology, the smoothing layer is equivalent to a channel-wise 1D convolutional layer with no bias and whose weights are fixed to the Gaussian kernel as calculated in Equation 4.11. It should also be noted here that as the transposed convolution operations used by S-TCNN explicitly exploits the spatial correlation along the frequency axis, it ensures the continuity of the predicted frequency responses. When combined with the learnable smoothing layer, the output of this block becomes a smooth and continuous response.

4.2.3 Causality Enforcement Layer (CEL)

The Kramers-Kroning relations given in Equation 4.5 suggest that in order to make the neural network generated S-Parameters represent a causal system, it is sufficient to only predict the real part and construct the imaginary part using the Hilbert transform. However, the S-Parameter data used for training is bandlimited and tabulated, leading to truncation and discretization errors that need to be accounted for. In other words, if we construct the imaginary part from the real part in the given frequency range, one can never achieve a zero reconstruction error. This is illustrated in Figure 4.2, where the reconstructed imaginary part of the return loss element begins to deviate from the actual imaginary part as the frequency approaches to the maximum available frequency.

In order to minimize the truncation error and maximize numerical accuracy of the neural network, we propose to extrapolate the real part of the S-Parameters. Let the broadband S-Parameters used for training data be written as

$$S_{ij}(f) = U_{ij}(f) + jV_{ij}(f), \;\; f \in \Omega = [f_{min}, f_{max}] \tag{4.13}$$

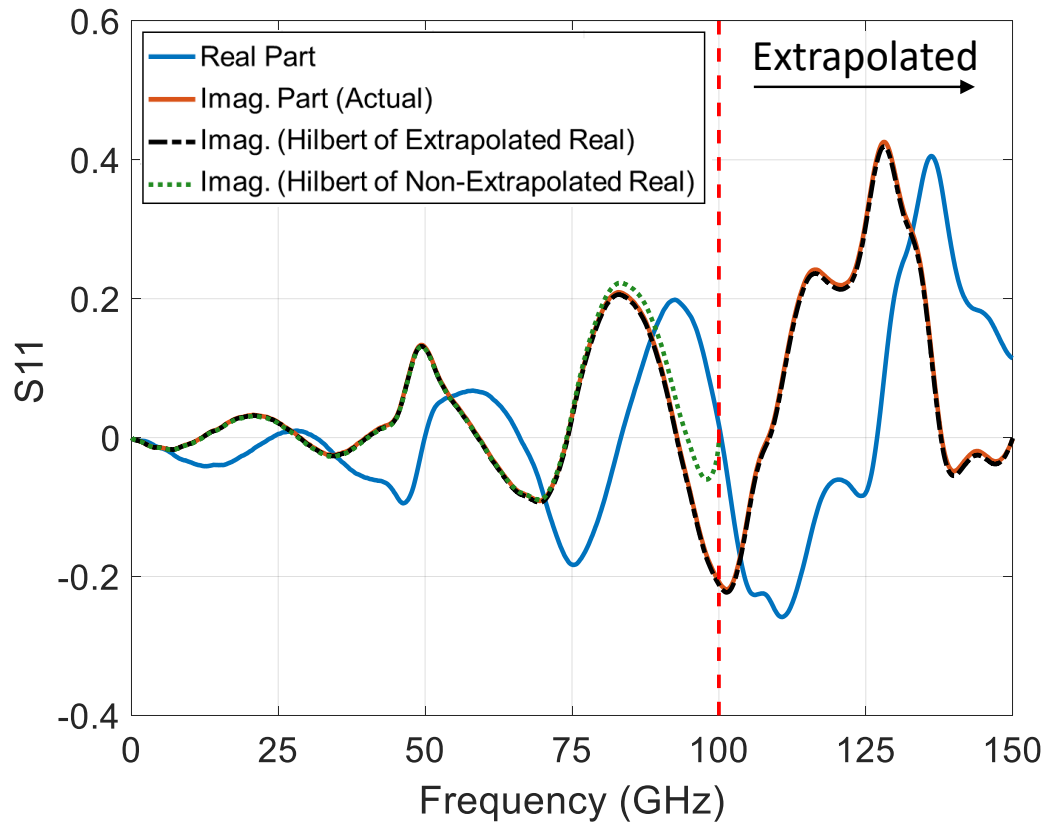


Figure 4.2: Illustration of the truncation error and the effect of extrapolation when reconstructing the imaginary part of return loss from the real part.

where $U(f)$ and $V(f)$ are the real and imaginary parts, respectively. The Hilbert transform integral in Equation 4.5 can then be split into in-band and out-of-band as

$$V(f) = \frac{-1}{\pi}\left[\int_{f\in|\Omega|} \frac{U(\widetilde{f})}{f-\widetilde{f}}\,\mathrm{d}\widetilde{f} + \int_{f\notin|\Omega|} \frac{U(\widetilde{f})}{f-\widetilde{f}}\,\mathrm{d}\widetilde{f}\right]. \tag{4.14}$$

The principal of analytic continuation [49] states that it is possible to find $U(f)$ for $f > f_{max}$ and $f < f_{min}$ since such information is contained within the observed part of the $V(f)$ in the form of out-of-band integration in Equation 4.14. As neural networks are universal approximators, it is possible to find an extrapolated response such that the in-band reconstruction has minimum error. This is demonstrated in Figure 4.2, where the extrapolated response found by the neural network minimizes the truncation error of the

conventional Hilbert transform.

Here, we exploit these properties and use the base network to extrapolate $U(f)$ until Mf_{max} where M is the extrapolation factor and backpropagate through in-band reconstruction error to minimize the effect of truncation error. In order to achieve this, we treat the $(NM + 1)$ point output of the base network, $\mathbf{y_{BN}}$ in Equation 4.10, as the extrapolated real part of the S-Parameter frequency response, where the addition of 1 represents the DC point. The input of the CEL is then the smoothened version of this quantity, $\mathbf{y_{SL}}$ in Equation 4.11.

As the Hilbert transform integral, and its derivatives with respect to the weights of the base network, is to be calculated during the training of the overall model, the direct numerical integration approach is not suitable due to the requirement of specialized integration kernels to handle singularities in the denominator. In addition, the direct integration needs to be evaluated at every discrete frequency point of every channel of $\mathbf{y_{SL}}$ for every training data, which can lead to significant computational overhead during both the training and inference of the neural network. Another well-known approach to compute the Hilbert transform is through forming the discrete analytical signal of the given sequence, $\mathbf{y_{SL}}$, through fast Fourier transform (FFT) [50]. In this chapter, we adopt the FFT based approach since it is highly computationally efficient as it allows for batched computation and is a differentiable operation. Note that FFT based computation effectively assumes the frequency domain signal to be periodic. However, the symmetricity of the real part of the frequency response indicates end-points of the dicrete sequence to be equal to each other. This eliminates the sharp discontinuity between the two consecutive periods of the frequency response that would otherwise cause ripple in the Fourier domain, thus, validates the periodicty assumption.

In particular, we first create the double-sided spectrum of $\mathbf{y_{SL}}$. As $\mathbf{y_{SL}}$ is of size $N_d \times D_y \times (NM + 1)$ where the third-axis represents the real part of the frequency spectrum,

this can be done by copying positive frequency points to the negative parts as

$$\widetilde{\mathbf{y}}_{\mathbf{SL}}^{(i,j,:)} = \left[\mathbf{y}_{\mathbf{SL}}^{(i,j)}[0, ..., NM],\ \ \mathbf{y}_{\mathbf{SL}}^{(i,j)}[NM-1, ..., 0]\right] \tag{4.15}$$

where $\widetilde{\mathbf{y}}_{\mathbf{SL}}$ is the double-sided spectrum of size $N_d \times D_y \times 2(NM+1)-1$ and the superscript $(i, j, :)$ denotes the operation is done in the third-axis for every (i, j) pair. Noting that $\widetilde{\mathbf{y}}_{\mathbf{SL}}$ is always an odd sized sequence along the frequency axis, we take FFT of $\widetilde{\mathbf{y}}_{\mathbf{SL}}$ along the third-axis to transform from frequency-domain into a new ν-domain and create the analytical part as

$$\widetilde{\mathbf{z}}_{\mathbf{SL}}^{(i,j,:)} = -\mathscr{H}\left\{\widetilde{\mathbf{y}}_{\mathbf{SL}}^{(i,j,:)}\right\} = -\operatorname{Im}\left\{\mathcal{F}^{-1}\left\{\mathbf{Z}^{(i,j)}[\nu]\right\}\right\} \tag{4.16}$$

where

$$\mathbf{Z}^{(i,j)}[\nu] = \begin{cases} \widetilde{\mathbf{Y}}^{(i,j)}[0], & \nu = 0 \\ 2\widetilde{\mathbf{Y}}^{(i,j)}[\nu], & 1 \leq \nu \leq (NM+1) \\ 0, & (NM+1) < \nu \leq 2(NM+1)-1 \end{cases} \tag{4.17}$$

and $\widetilde{\mathbf{Y}}^{(i,j)}[v] = \mathcal{F}\left\{\widetilde{\mathbf{y}}_{\mathbf{SL}}^{(i,j,:)}\right\}$ is the discrete Fourier transform of $\widetilde{\mathbf{y}}_{\mathbf{SL}}^{(i,j)}$ in the ν-domain and the superscript $(i, j, :)$ denotes the operation is done along the third-axis. The operation in Equation 4.17 corresponds to creating the discrete analytic signal of $\widetilde{\mathbf{y}}_{\mathbf{SL}}$, which is one-sided in ν-domain as $\mathbf{Z}^{(i,j)}[\nu] = 0$ for $\nu < 0$. After getting rid of redundant negative frequency points, i.e. $n > (NM+1)$, the resulting $\widetilde{\mathbf{z}}_{\mathbf{SL}}$ is of size $N_d \times D_y \times (NM+1)$, which represents the imaginary part of S-Parameter response at a frequency step size of f_s for every data point.

In order to address the discretization error, f_s can be arbitrarily decreased at this step of the neural network by a factor of K by interpolating both $\mathbf{y}_{\mathbf{SL}}$ and $\mathbf{z}_{\mathbf{SL}}$. However, such an interpolation can disrupt the analytic behavior of the output and result in a non-causal response. Here, to preserve the analytic behavior of the output, the interpolation should

preserve the orthogonality of $\mathbf{y_{SL}}$ and $\mathbf{z_{SL}}$, written as

$$\sum_{n=0}^{NM} \mathbf{y}_{\mathbf{SL}}^{(i,j)}[n]\mathbf{z}_{\mathbf{SL}}^{(i,j)}[n] = 0. \tag{4.18}$$

Since the spectrum of $\mathbf{Z}^{(i,j)}[\nu]$ is one-sided, one can arbitrarily pad zeros to $\mathbf{Z}^{(i,j)}[\nu]$ for $\nu < 0$, which would preserve the orthogonality condition in Equation 4.18 while interpolating $\mathbf{y_{SL}}$ and $\mathbf{z_{SL}}$ [50]. In order to reduce the frequency step size to f_s/K, the procedure in Equation 4.16 and Equation 4.17 can be re-written as

$$\begin{aligned} \widetilde{\mathbf{y}}_{\mathbf{CEL}}^{(i,j,:)} &= K \operatorname{Re}\left\{\mathcal{F}^{-1}\left\{\mathbf{Z}^{(i,j)}[\nu]\right\}\right\} \\ \widetilde{\mathbf{z}}_{\mathbf{CEL}}^{(i,j,:)} &= -K \operatorname{Im}\left\{\mathcal{F}^{-1}\left\{\mathbf{Z}^{(i,j)}[\nu]\right\}\right\} \end{aligned} \tag{4.19}$$

where

$$\mathbf{Z}^{(i,j)}[\nu] = \begin{cases} \widetilde{\mathbf{Y}}^{(i,j)}[0], & \nu = 0 \\ 2\widetilde{\mathbf{Y}}^{(i,j)}[\nu], & 1 \leq \nu \leq (NM+1) \\ 0, & (NM+1) < \nu \leq \left(2NMK + K\right) \end{cases} \tag{4.20}$$

After the analytic signal is formed, we truncate both $\widetilde{\mathbf{y}}_{\mathbf{CEL}}^{(i,j,:)}$ and $\widetilde{\mathbf{z}}_{\mathbf{CEL}}^{(i,j,:)}$ for $n > (NK+1)$ to get rid of the extrapolated part since the out-of-band predictions are not of interest in this thesis. The overall block-diagram of CEL is given in Figure 4.3, which can be written as a parameterless NN layer that takes a real tensor of size $N_d \times D_y \times (NM+1)$ as input and constructs a complex output that is the frequency response of a causal system with a frequency step size of f_s/K as

$$\mathbf{S_{CEL}}^{(i,j)}[n] = \widetilde{\mathbf{y}}_{\mathbf{CEL}}^{(i,j)}[n] + j\ \widetilde{\mathbf{z}}_{\mathbf{CEL}}^{(i,j)}[n] \tag{4.21}$$

where $\widetilde{\mathbf{y}}_{\mathbf{CEL}}^{(i,j,:)}$ and $\widetilde{\mathbf{z}}_{\mathbf{CEL}}^{(i,j,:)}$ are calculated as in Equation 4.19 and Equation 4.20 and truncated, $\mathbf{S_{CEL}}$ is the $N_d \times D_y \times (NK+1)$ sized complex output tensor of the CEL, which is forwarded to PEL. It should be noted that K and M are hyperparameters of the overall

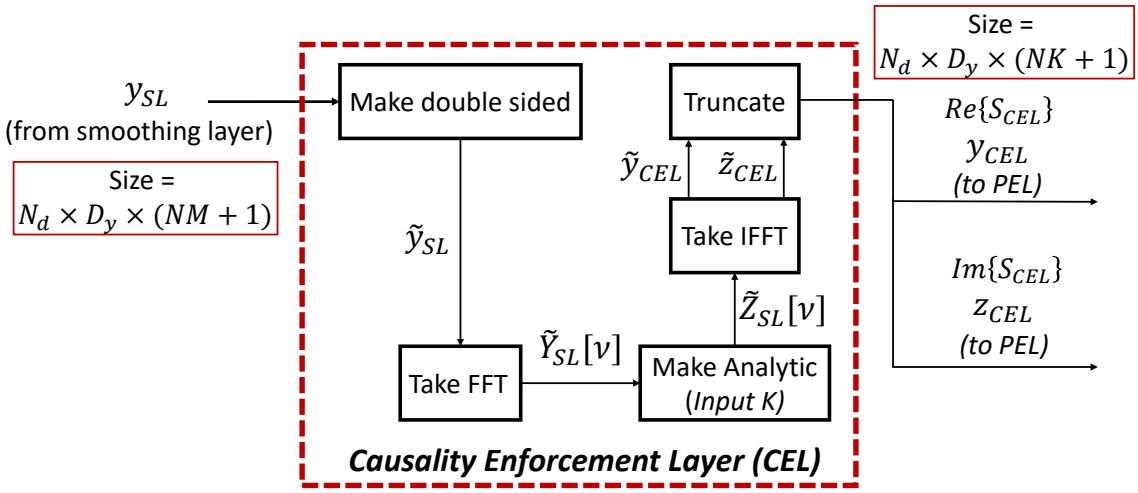


Figure 4.3: Block-diagram summary of the operations done in CEL.

network architecture, just like the number of layers and neurons in a regular NN, and should be chosen based on the application.

4.2.4 Passivity Enforcement Layer (PEL)

As explained in detail in Section 8.1, the neural network predicted S-Parameters are passive if and only if the maximum singular value condition in Equation 4.9 is satisfied. In order to check this condition, the flattened complex S-Parameter representation, $\mathbf{S_{CEL}}$, of size $N_d \times D_y \times (NK+1)$ should be re-shaped into the batched matrix form of $N_d \times P \times P \times (NK+1)$ for a P-port network, where the first and last dimensions are number of data and frequency points, respectively.

Since the re-shaped $\mathbf{S_{CEL}}$ matrices are complex-valued and neural network training and inference with complex values are not possible, we use isomorphism [51] to transform $\mathbf{S_{CEL}}$ as

$$\mathbf{S_P} = \begin{bmatrix} [1.5]\,\mathrm{Re}\{\mathbf{S_{CEL}}\} & \mathrm{Im}\{\mathbf{S_{CEL}}\} \\ -\mathrm{Im}\{\mathbf{S_{CEL}}\} & \mathrm{Re}\{\mathbf{S_{CEL}}\} \end{bmatrix} \tag{4.22}$$

where $\mathbf{S_P}$ is of size $N_d \times 2P \times 2P \times (NK+1)$. This representation allows to use regular neural network computations and other linear algebraic operations for complex valued matrices. In order to check and enforce passivity to $\mathbf{S_P}$, one needs to determine its maxi-

mum singular value, σ_1, at every data and frequency point. However, most efficient SVD algorithms contain sequential bidiagonalization operations and can not be efficiently parallelized. This would add a significant computational overhead to the training of the neural network since in order to characterize passivity of every $2P \times 2P$ matrix in $\mathbf{S_P}$, one needs to perform $N_d(NK+1)$ sequential SVD and gradient operations at every iteration of the training process.

To limit the computational overhead, we propose to use an upper bound to σ_1 that can be calculated using only matrix-matrix multiplications and Hadamard products, hence, can be massively parallelized [52]. Let

$$\begin{aligned} C(f) &= tr\left(S^H(f)S(f)\right) \\ D(f) &= tr\big(\left(S^H(f)S(f)\right)^2\big) \end{aligned} \tag{4.23}$$

where $tr(\cdot)$ is the trace operator. An upper bound to maximum singular value of a $P \times P$ S-Parameter matrix can then be calculated as

$$\sigma_1(f) \leq \hat{\sigma}_1(f)$$

where

$$\hat{\sigma}_1(f) = \sqrt{\frac{C(f)}{P} + \left(\frac{P-1}{P}\left(D(f) - \frac{C(f)^2}{P}\right)\right)^{0.5}}. \tag{4.24}$$

To minimize the number of matrix-matrix multiplications that has greater than quadratic time complexity, $C(f)$ and $D(f)$ can be calculated efficiently using Hadamard products as

$$C(f) = \sum_{i=1}^{P} |S_{ii}(f)|^2 \tag{4.25}$$

$$D(f) = \sum_{i,j=1}^{P} \Big[\left(S^H(f)S(f)\right) \circ \left(S(f)S^H(f)\right) \Big]_{ij} \tag{4.26}$$

where $\circ$ is the Hadamard product. The proof of the upper bound follows from the bounds

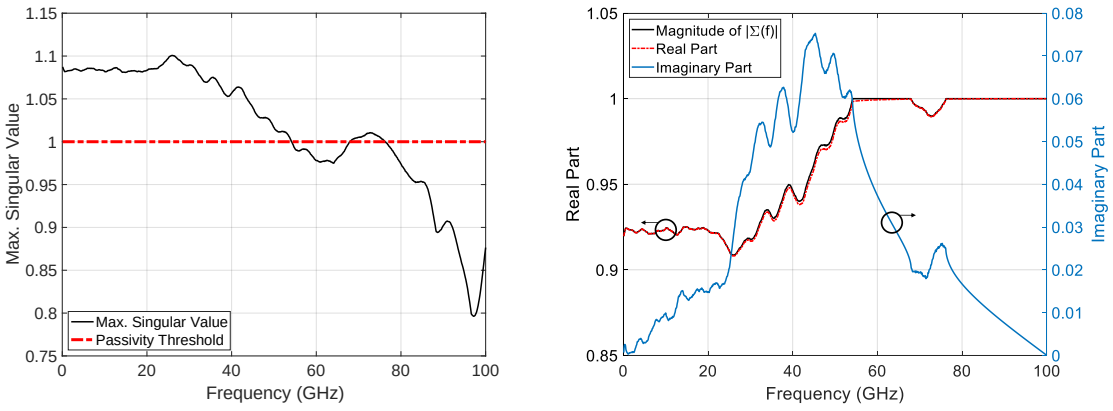
for eigenvalues derived using a Cauchy-Schwarz type inequality [53] and the fact that $\sigma_1(f)$ is the square root of the largest eigenvalue of $S^H(f)S(f)$. Experiments and further details regarding the tightness of this bound for various types of matrices can be found in [52]. For typical S-Parameter matrices of interest in this chapter, the average relative deviation between the maximum singular value and its upper bound is found to be approximately 3%. We further emphasize here that Equation 4.25 and Equation 4.26, thereby Equation 4.24, are simple multiplication and accumulation operations, hence, can be easily parallelized both during the training and inference.

Once the $\hat{\sigma}_1(f)$ is obtained, there are several ways to enforce passivity to every matrix in $\mathbf{S_P}$. One way is to find $\hat{\sigma}_{\text{max}} = \max(\hat{\sigma}_1(f))$ for every data point and divide the whole frequency spectrum of each element in $\mathbf{S_{CEL}}$ by $\hat{\sigma}_{\text{max}}$ if it is greater than 1 [54]. Note that this enforcement would increase the numerical error between the predicted and actual S-Parameter responses. As this enforcement is to be done during the training of the network, weights and biases of the base network are automatically adjusted to minimize these deviations. However, we have observed that, in practice, this results in over-reducing $\sigma_1(f)$ near DC point, where losses are minimal.

Another approach is to perform frequency-dependent enforcement, i.e. divide $\mathbf{S_P}(f)$ by $\sigma_1(f)$ if passivity is violated. This is shown to be the optimal enforcement in frequency domain in the sense that $\mathbf{S_P}(f)$ is minimally perturbed at every frequency point [55]. However, frequency-dependent enforcement results in disrupting Kramers-Kronig relations, hence, the resulting S-Parameters become non-causal.

In this chapter, we propose a new causality-preserving passivity enforcement technique that can be utilized in a neural network environment. We view the frequency-dependent passivity enforcement operation as filtering in the frequency domain, written as

$$\mathbf{S_{PEL}}(f) = \mathbf{S_{CEL}}(f) \odot \mathbf{\Sigma}(f) \tag{4.27}$$



(a) (b)

Figure 4.4: Illustration for constructing the proposed minimum-phase passivity enforcement filter. (a) Singular value to be filtered. (b) Magnitude, real and imaginary parts of the filter.

where $\mathbf{\Sigma}(f)$ is the complex-valued passivity enforcement filter of size $N_d \times (NM+1)$ and $\odot$ operator is used to indicate the filtering is done along the frequency axis of each channel of $\mathbf{S_{CEL}}$ at every data point. In order to inflict minimal changes to $\mathbf{S_{CEL}}(f)$, the desired magnitude spectrum of $\mathbf{\Sigma}(f)$ at each data point can be written as

$$\big|\Sigma(f)\big| = \begin{cases} \dfrac{1}{\hat{\sigma}_1(f)} & \text{, for } \hat{\sigma}_1(f) > 1 \\ 1 & \text{, for } \hat{\sigma}_1(f) \leq 1. \end{cases} \tag{4.28}$$

It has been shown that if $\Sigma(f)$ is a minimum-phase filter and $\mathbf{S_{CEL}}$ is the frequency response of a causal function, $\mathbf{S_{PEL}}$ also represents the frequency response of a causal system [42].

A minimum-phase frequency response can be formed solely from its magnitude spectrum [56]. Here, we propose to exploit these to construct a minimum-phase passivity enforcement filter from the desired magnitude spectrum in Equation 4.28 as

$$\begin{aligned} \Sigma(f) &= \big|\Sigma(f)\big| e^{j\theta(f)} \\ \theta(f) &= \mathscr{H}\big\{ \log|\Sigma(f)| \big\} \end{aligned} \tag{4.29}$$

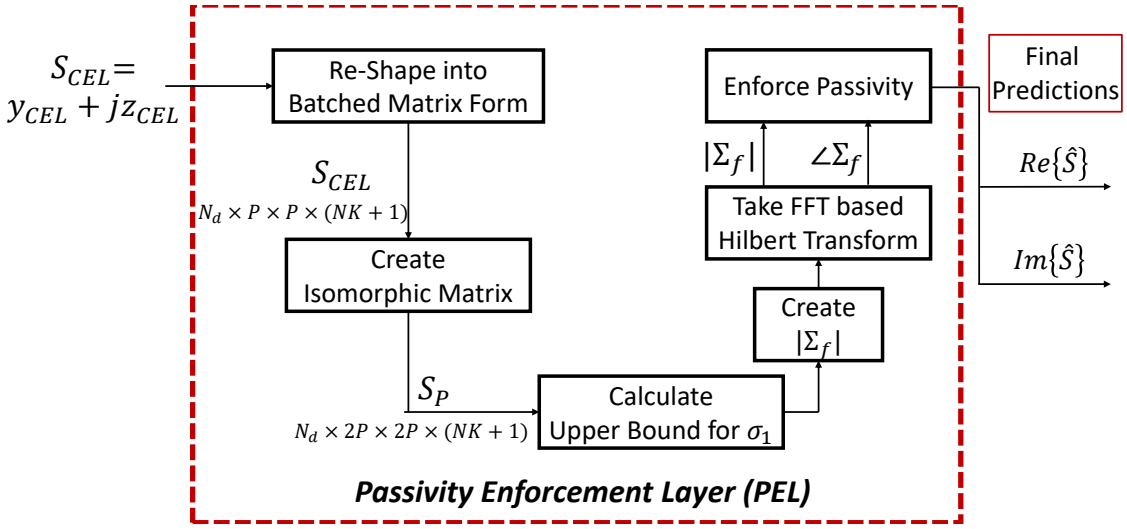


Figure 4.5: Block-diagram summary of the operations done in PEL.

where $\log(\cdot)$ is the natural logarithm operator and the Hilbert transform is taken using the FFT based approach outlined in Equation 4.15-Equation 4.17. An example of a complex-valued passivity enforcement filter constructed using the proposed approach is also given in Figure 4.4.

Overall operations that PEL performs are summarized in Figure 4.5. Similar to CEL, PEL can be written as a parameterless neural network layer that takes $N_d \times D_y \times (NK+1)$ complex tensor as input and constructs a complex tensor that is the final predicted response as

$$\begin{aligned}\hat{\mathbf{S}}[n] &= f\left(\mathbf{S_{CEL}}[n]\right) \\ &= \mathbf{S_{CEL}}[n] \odot \mathbf{\Sigma}[n], \quad n = 0, ..., NK\end{aligned} \tag{4.30}$$

where $\hat{S}[n]$ is the $N_d \times D_y \times (NK+1)$ sized complex output tensor that represent the predicted broadband S-Parameters of a causal and passive system.

4.2.5 Training Methodology

The training of the proposed network architecture can be performed using the conventional backpropagation method since all the operations are differentiable. Although CEL and PEL are both comprised of computationally efficient and parallelizable operations, batched matrix calculations in Equation 4.24-Equation 4.26 can create some computational overhead during the training.

In order to further minimize this overhead, we propose using a two-step training methodology. As the S-Parameters used for training data are passive at all frequencies, minimization of the training error corresponds to making the singular values of the predicted S-Parameters close to their actual values during the training process. Exploiting this, for the first L_f iterations, i.e. gradient updates, of the total L iterations allocated for the training process, we bypass PEL and directly compare $\mathbf{S_{CEL}}$ with the training data to calculate the error. We then activate PEL to guarantee passivity, causing a slight jump in the training error due to non-passivities, which is then minimized for the remaining $(L - L_f)$ iterations. For the application examples given in Section IV-VI, we have found that this results in almost identical training and test error, but with a lower computational time used for the training.

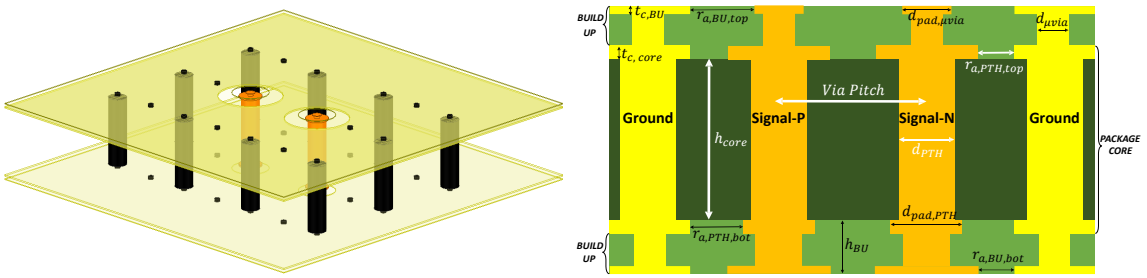
As for the training error metric, instead of the conventional mean-squared error (MSE) loss, we propose using the modified loss function that is more suitable for predicting frequency responses as in Chapter 7, which is repeated here for convenience as

$$L = \sqrt{\frac{1}{N_d D_y} \sum_{n=1}^{N_d} \sum_{d=1}^{D_y} L_f^{(n,d)}} \tag{4.31}$$

where

$$L_f^{(n,d)} = \sqrt{\frac{1}{N} \sum_{m=0}^{N} \left(S^{(n,d)}[m] - \hat{S}^{(n,d)}[mK] \right)^2}$$

and $S^{(n,d)}[m]$ is the frequency response for the n^{th} data point of the d^{th} channel of the



(a) (b)

Figure 4.6: Geometry of the differential PTH structure in package core. (a) Isometric view. (b) Cross-section.

training S-Parameters at m^{th} frequency point, and the factor of K in the predicted response, $\hat{S}^{(n,d)}[mK]$, comes from Equation 4.19-Equation 4.20 where frequency step size of the predictions are reduced by a factor of K. The loss function in Equation 4.31 is based on the scaled ℓ^2-norm of the error between the predicted and the actual frequency responses, averaged over $N_d \times D_y$ different frequency responses in the training set.

4.3 Application 1: Differential PTH Pair in Package Core

The first application chosen to demonstrate the proposed method is modeling a differential PTH pair in package core along with the microvias that connect immediate build-up (BU) layers to the package core. Such structures are common in off-chip high-speed channels and often times can be the bottleneck that limits the total channel bandwidth. Hence, it is important to optimize its design to minimize distortions in the signal. The objective here then is to learn a causal and passive mapping from the geometrical parameters of the PTH structure to 4-port single ended broadband S-Parameters, such that it can be used later in time-domain simulations to capture its effect on high-speed signaling. The input parameters comprise a 13 dimensional space and their corresponding bounds are given in Figure 4.6 and Table 4.1. Note that the large sample space is chosen to contain various technology nodes to avoid creating different models for different manufacturing technologies.

Table 4.1: Control Parameters of the PTH structure

Parameter		Unit	Min	Max
μ-via Diameter	$d_{\mu\text{-via}}$	μm	30	70
μ-via Pad Diameter	$d_{\text{pad, }\mu\text{-via}}$	μm	31	140
BU Layer Thickness	h_{BU}	μm	20	35
μ-via Top Antipad Radius	$r_{\text{a, BU, TOP}}$	μm	100	500
μ-via Bot. Antipad Radius	$r_{\text{a, BU, BOT}}$	μm	100	500
PTH Pitch	v_p	μm	300	1200
Core Thickness	h_{Core}	μm	100	1200
BU Copper Thickness	$t_{\text{c, BU}}$	μm	10	20
Core Copper Thickness	$t_{\text{c, CORE}}$	μm	11	40
PTH Diameter	d_{PTH}	μm	100	250
PTH Pad Diameter	$d_{\text{pad, PTH}}$	μm	110	500
PTH Top Antipad Radius	$r_{\text{a, PTH, TOP}}$	μm	50	500
PTH Bot. Antipad Radius	$r_{\text{a, PTH, BOT}}$	μm	50	500

4.3.1 Simulation and Model Setup

In order to create the predictive model, 680 samples based on Latin Hypbercube Sampling (LHS) are determined. These are then fed into a full-wave EM solver to generate their corresponding S-Parameters between 0.1-100 GHz at 100 MHz frequency steps, where the structure is excited using coaxial waveports at the antipads of microvias. After the data is collected, 550 out of 680 samples are used for the training of the model. As the PTH structure is a partially symmetric and reciprocal system, the target data to be trained, i.e. output channels of the model, is determined to be the real and imaginary parts of the frequency responses of S_{11}, S_{12}, S_{13}, S_{14}, S_{33} and S_{34}, corresponding to a total of 12,000 output dimensions. We use $K = 2$ in Equation 4.19 and $M = 1.5$ in Equation 4.17 to reduce the frequency step size of the predicted response to 50 MHz. The model then predicts the S-Parameters from DC up to 100 GHz at 50 MHz steps as the extrapolated part is truncated.

4.3.2 Results

We compare the proposed methodology (S-TCNN + CEL + PEL) with S-TCNN that also uses the learnable smoothing layer and a deep fully-connected neural network (DNN) that considers frequency as an additional input parameter to the model. The metric to assess the numerical accuracy for both models is chosen as the normalized mean squared error (NMSE) over each frequency response in the test set, given as

$$NMSE = \frac{1}{N_d D_y} \times \sum_{d=1}^{D_y} \sum_{n=1}^{N_d} \left(\frac{\sum_{m=1}^{N} \left(S_{n,d}[m] - \hat{S}_{n,d}[mK] \right)^2}{\sum_{m=1}^{N} \left(S_{n,d}[m] - \frac{1}{N} \sum_{m=1}^{N} S_{n,d}[m] \right)^2} \right). \tag{4.32}$$

where $N_d = 130$ is the number of validation data, and $D_y = 12$ represent real and imaginary part of the learned S-Parameters.

Table 4.2 summarizes the NMSE values for each model along with the causality metrics and passivity violations of the predicted S-Parameters that are calculated using a commercial tool. In order to assess the predictive accuracy, we trained each model 10 times using randomly initialized weights and report mean and standard deviation of the NMSE values in Table 4.2. The DNN model performed significantly worse as compared to the other models and had an average NMSE of 9.42%, whereas S-TCNN and S-TCNN+CEL+PEL models performed similarly with an average NMSE of 5.08% and 5.34%, respectively. Predicted real and imaginary parts of differential insertion (IL) and return loss (RL) terms for different test cases are also shown in Figure 4.7.

In terms of physical consistency, S-Parameters predicted by S-TCNN and DNN had average causality quality metrics of 11.17% and 7.49%, respectively, and none of the S-Parameters predicted by either model were causal, whereas all the S-Parameters predicted by the proposed model were found to be causal with a quality metric of 100.0%. We further characterize the maximum singular values of S-Parameters predicted by each model for every case in the test set. We observe that S-TCNN+CEL+PEL guarantees passivity

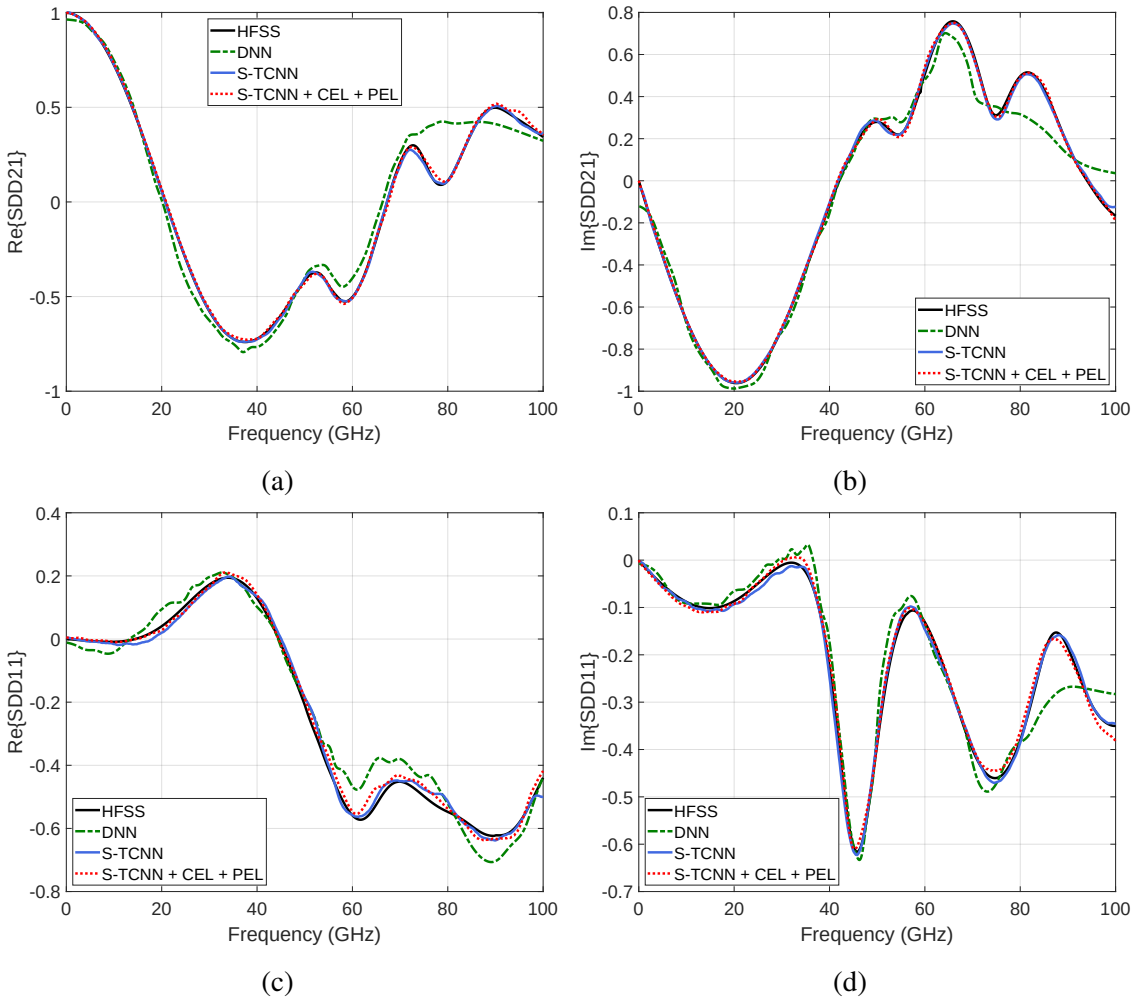


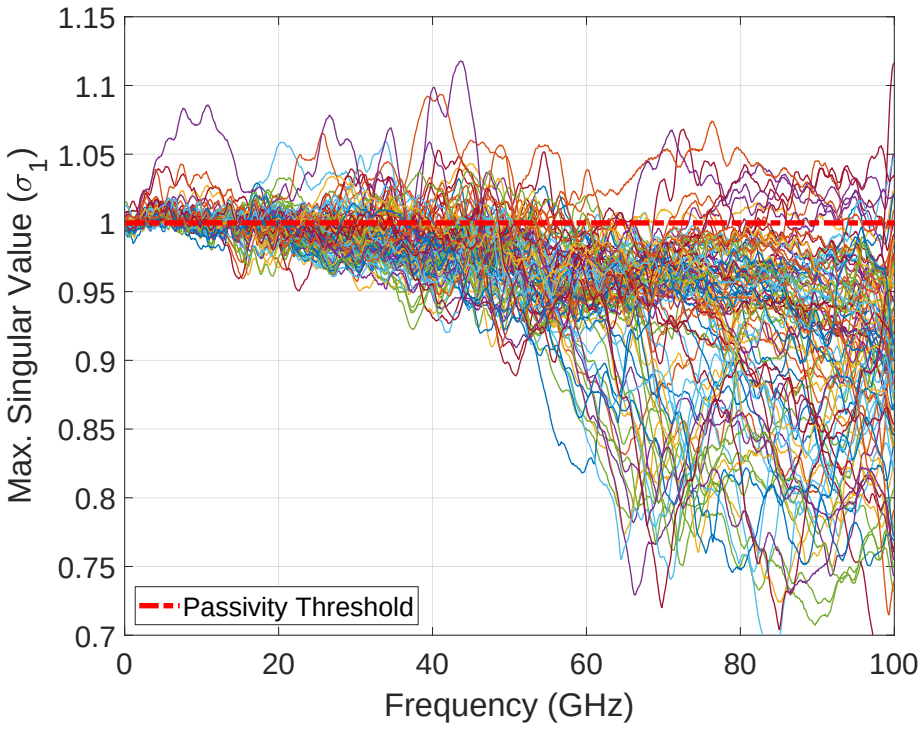
Figure 4.7: Comparison of predicted S-Parameters with 3D EM simulations for different test cases of PTH model. (a, b) Real and imaginary part of differential insertion loss. (c, d) Real and imaginary part of differential return loss.

Table 4.2: Comparison of Models on Test Data for PTH Model

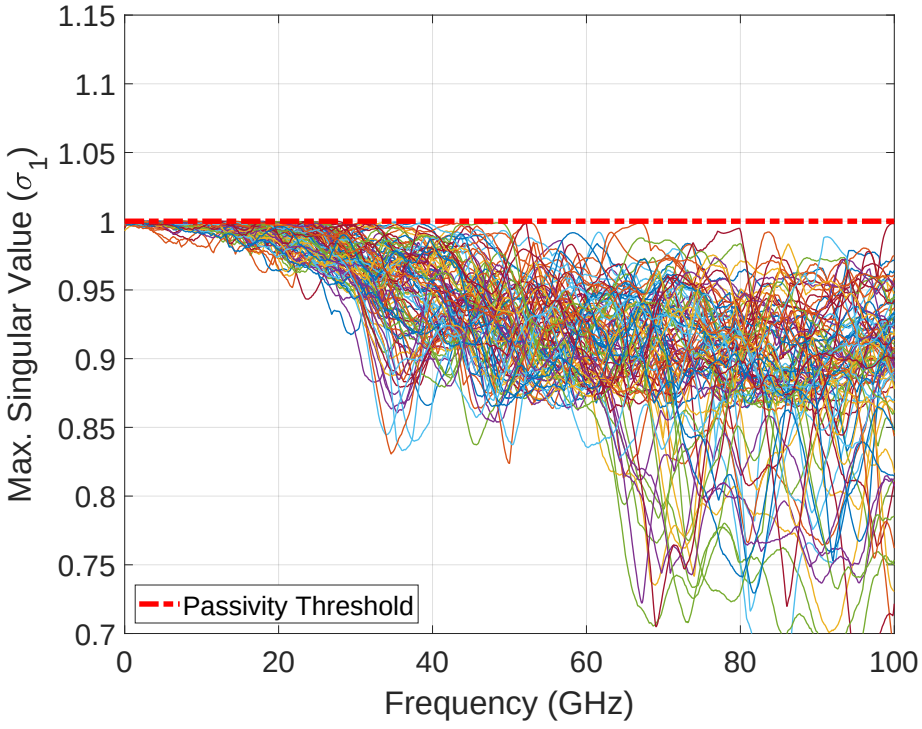
	DNN	**S-TCNN**	**S-TCNN + CEL + PEL**
NMSE	9.42% $\pm$ 0.33%	5.08% $\pm$ 0.31%	5.34% $\pm$ 0.09%
Av. Causality Metric	7.49%	11.17%	100.0%
Range of σ_1	[0.615, 1.285]	[0.653, 1.131]	[0.607, 0.999]
Inference Time (for 1K broadband S-Params.)	0.26 s	0.13 s	0.91 s

of the predicted S-Parameters as the σ_1 of all predictions are less than 0.99, while the predictions done by using S-TCNN and DNN do not necessarily result in a passive response as most of the predicted responses have $\sigma_{\max} > 1$. The maximum singular values of the predicted S-matrices for S-TCNN and S-TCNN+CEL+PEL model are further given in Figure 4.8 to show the effect of PEL on the passivity. In terms of inference run times, it took 0.13s for S-TCNN model to generate 1000 frequency responses as compared to 0.91s for S-TCNN+CEL+PEL and 0.26s for DNN, showing the operations done in CEL and PEL have minimal computational overhead to the overall model. Note that for a fair comparison, we have tuned the hyperparameters of all the models to achieve highest prediction accuracy. The best performing DNN model had 132,632 learnable parameters and consisted of three hidden layers with 250 neurons in each layer (14-250-250-250-12). The S-TCNN model had 43,048 learnable parameters and consisted of two fully-connected layers with 30 neurons (13-30-30), followed by five 1D transposed convolutional layers, each having 30 channels. The kernel size and stride for these layers were 32, 4, 4, 4, 2 and 1, 2, 2, 4 and 2, respectively. The base network of the S-TCNN+CEL+PEL model had the same S-TCNN architecture except a stride of 3 was used for the last layer. We have implemented each model using PyTorch [1] and for all the models, we used exponential linear units (ELUs) [57] as the activation function. The training was performed using the Adam optimizer [58] with an initial learning rate (LR) of 0.01 while reducing LR by $\times 0.5$ at every 500 iterations for a total of 3000 gradient updates. We have used the same model settings for the applications in the following sections.

To show the significance of physical consistency of the predicted S-Parameters, we use the predictions of the two best performing models (S-TCNN and S-TCNN+CEL+PEL) in a subsequent time-domain characterization and compare with the data obtained from 3D EM simulations. We perform a differential time-domain reflectometry (TDR) and transmission (TDT) using a 15 ps 0-100% rise-time cosine-edge step and pulse as differential excitation, respectively. These are common time-domain characterization techniques for high-speed

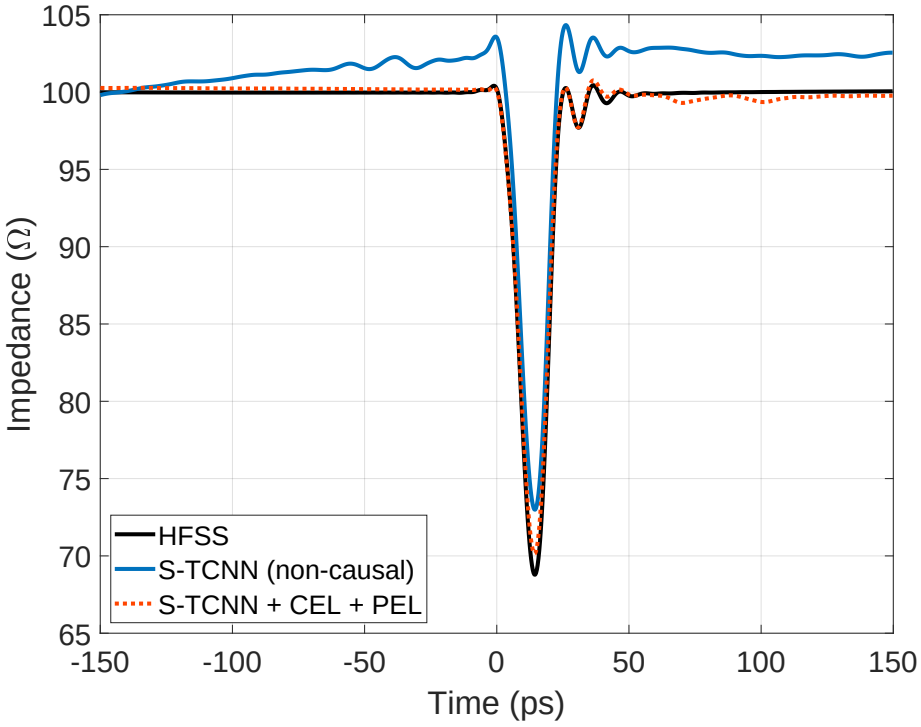


(a)

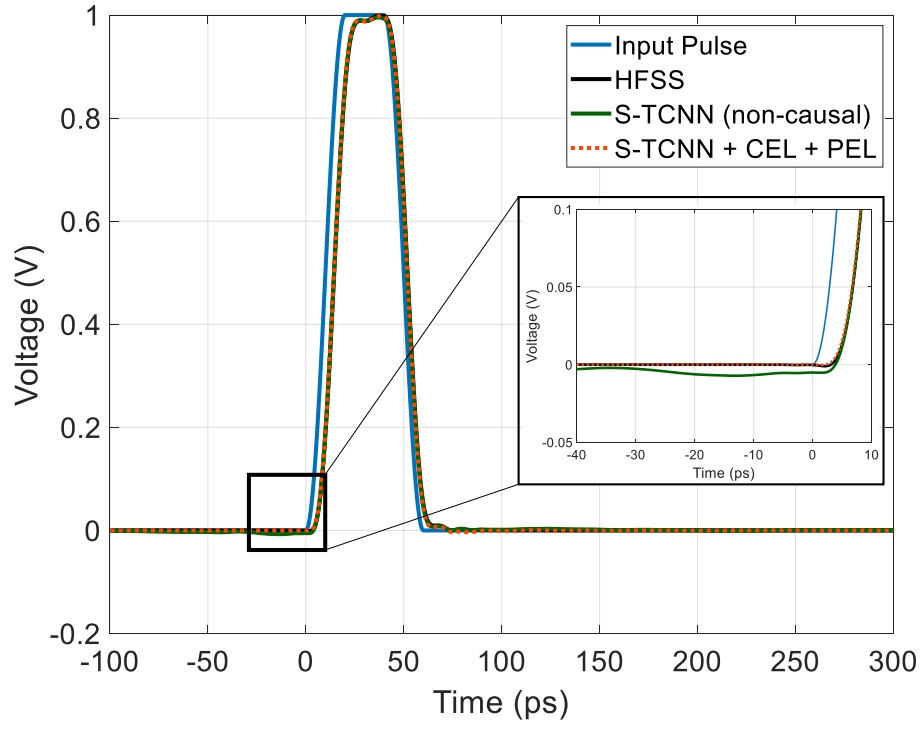


(b)

Figure 4.8: Passivity characterization of the predicted S-Parameters for every case in the test set. (a) S-TCNN. (b) S-TCNN + CEL + PEL.



(a)



(b)

Figure 4.9: Time-domain characterization of predicted S-Parameters for excitations with 15 ps 0-100% cosine-edge rise time. (a) TDR. (b) TDT.

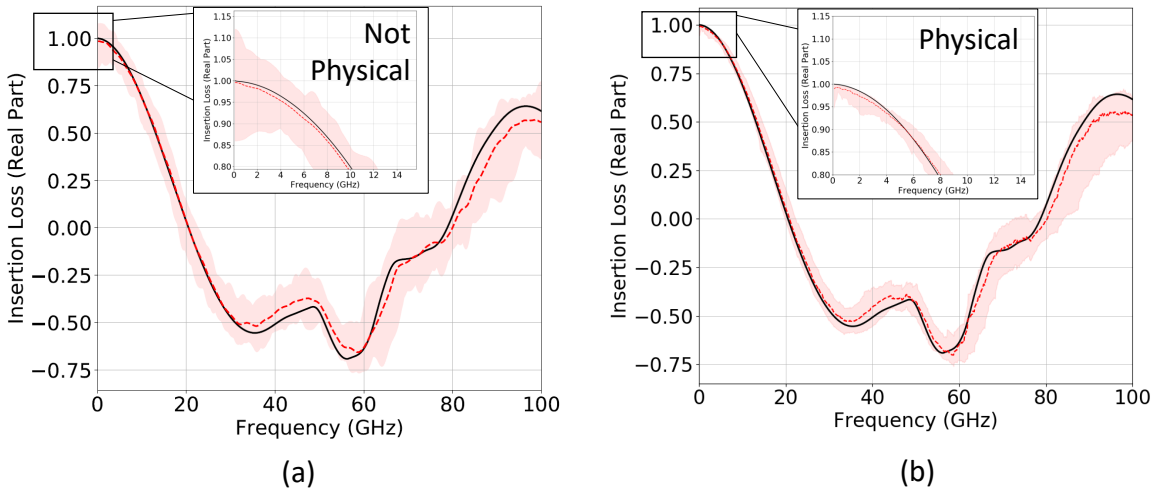


Figure 4.10: Confidence bounds obtained with Bayesian S-TCNN on PTH application with and without CEL and PEL, showing that the proposed CEL and PEL enables learning a *physically consistent posterior distribution*. (a) Bayesian S-TCNN. (b) Bayesian S-TCNN + CEL + PEL.

channel analysis and their quality is of utmost importance to make design choices. The results of these characterizations are given in Figure 4.9. The TDR simulation in Figure 4.9a clearly shows that S-Parameters predicted by the S-TCNN are non-causal and significantly distort the characterization since a substantial impedance change can be observed before the structure is excited at $t = 0$, which is not present for S-Parameters predicted by the proposed model. Similar behavior is also observed for the TDT simulation in Figure 4.9b, where a substantial input power is leaked before the excitation, which can cause non-realistic intersymbol interference (ISI) when a long bit pattern is simulated. The results clearly show that although predicted S-Parameters have the same level of numerical accuracy with respect to the 3D EM simulation, pure neural network predicted S-Parameters are not physically consistent and can not be used in subsequent simulations.

Last but not least, we train two additional Bayesian models, Bayesian S-TCNN and Bayesian S-TCNN + CEL + PEL, using the MC dropout technique presented in Section 7.3 to show the effect of CEL and PEL on confidence bounds on S-Parameter predictions. As can be seen in Figure 4.10, the confidence bounds obtained by Bayesian S-TCNN model violates the physical consistency since the real part of differential insertion loss exceeds

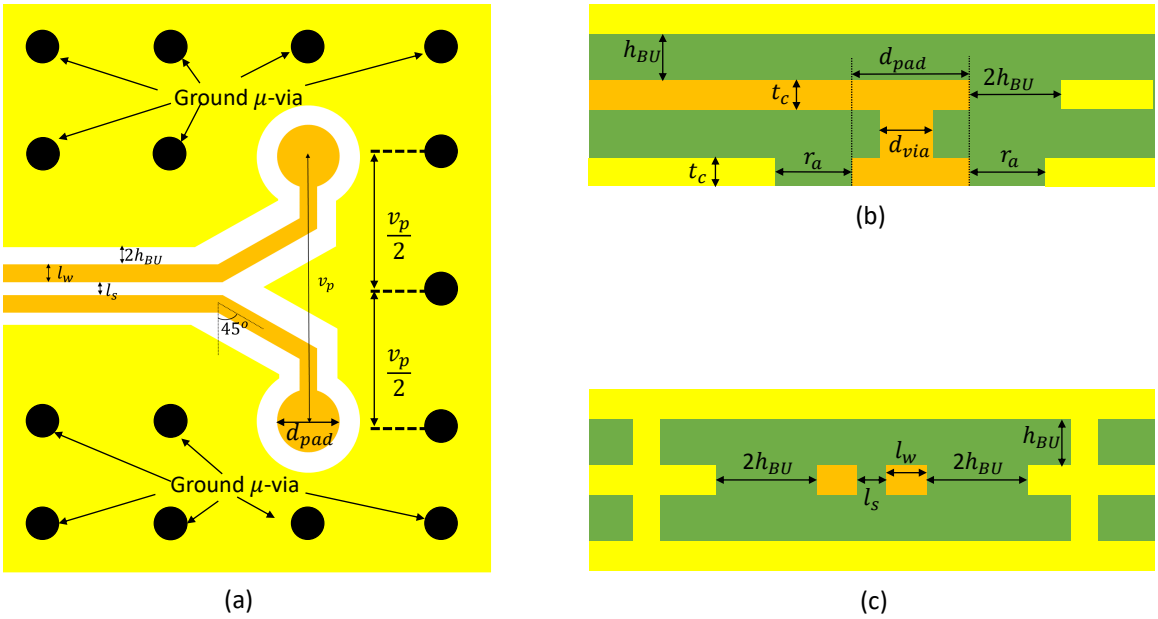


Figure 4.11: Geometry of the differential stripline pair in package. (a) Top view. (b) Front view. (c) Side view.

1. On the other, the confidence bounds obtained through Bayesian S-TCNN + CEL + PEL model is physically consistent and no violations are observed. This further shows the effectiveness of CEL and PEL in Bayesian modeling in the sense that the model is able to learn a *physically consistent distribution* such that every sample from the learned posterior becomes causal and passive frequency responses. This property of Bayesian S-TCNN + CEL + PEL makes it applicable to uncertainty quantification problems where the effect of process variations needs to be observed on both time- and frequency-domains, which was not possible with GP models presented in Chapter 6.

4.4 Application 2: Differential Stripline Pair in Package

The second application we choose in this chapter is parameterizing the frequency response of a differential stripline structure in package. Such transmission lines are commonly a part of the chip-to-board escape route in high-speed channels, and their impedance can greatly affect the achievable communication bandwidth. Since their return path and transition to microvias are non-uniform, they need to be characterized using computationally

Table 4.3: Control Parameters of the stripline structure.

Parameter		Unit	Min	Max
Line Width	l_w	μm	15	75
Pair Spacing	l_s	μm	30	60
μ-via Diameter	d_{via}	μm	30	70
μ-via Pad Diameter	d_{pad}	μm	31	140
μ-via Antipad Radius	r_a	μm	50	500
Via Pitch	v_p	μm	300	1200
Copper Thickness	t_c	μm	10	20
BU Layer Thickness	h_{BU}	μm	20	35

expensive 3D EM simulations. In this section, we therefore derive a parametric model to predict 4-port single-ended S-Parameters such that it can be later used for detailed parametric analysis. The geometry of the differential stripline pair is given in Figure 4.11 and is parameterized using 8 parameters as in Table 4.3.

4.4.1 Simulation Setup

Similar to the PTH application in the previous section, we determine 940 samples based on LHS and simulate broadband S-Parameters for each sample between 0.1-100 GHz at 100 MHz steps, where the ports are defined at front of the stripline and antipads of the microvias. 750 of 940 samples are used for training, and the rest is used for validation. The output parameters, output dimensionality and settings of the model are taken as the same as described in subsection subsection 4.3.1.

4.4.2 Results

The results are summarized in Table 4.4. The DNN model had an average NMSE of 4.73%, as compared to 3.08% and 2.47% for S-TCNN and S-TCNN+CEL+PEL, respectively. Figure 4.12 further compares the predicted differential insertion and return losses to 3D EM simulations. It can be seen that the convolutional type models captured both smooth and resonant parts of the frequency response, whereas the DNN model had lower accuracy

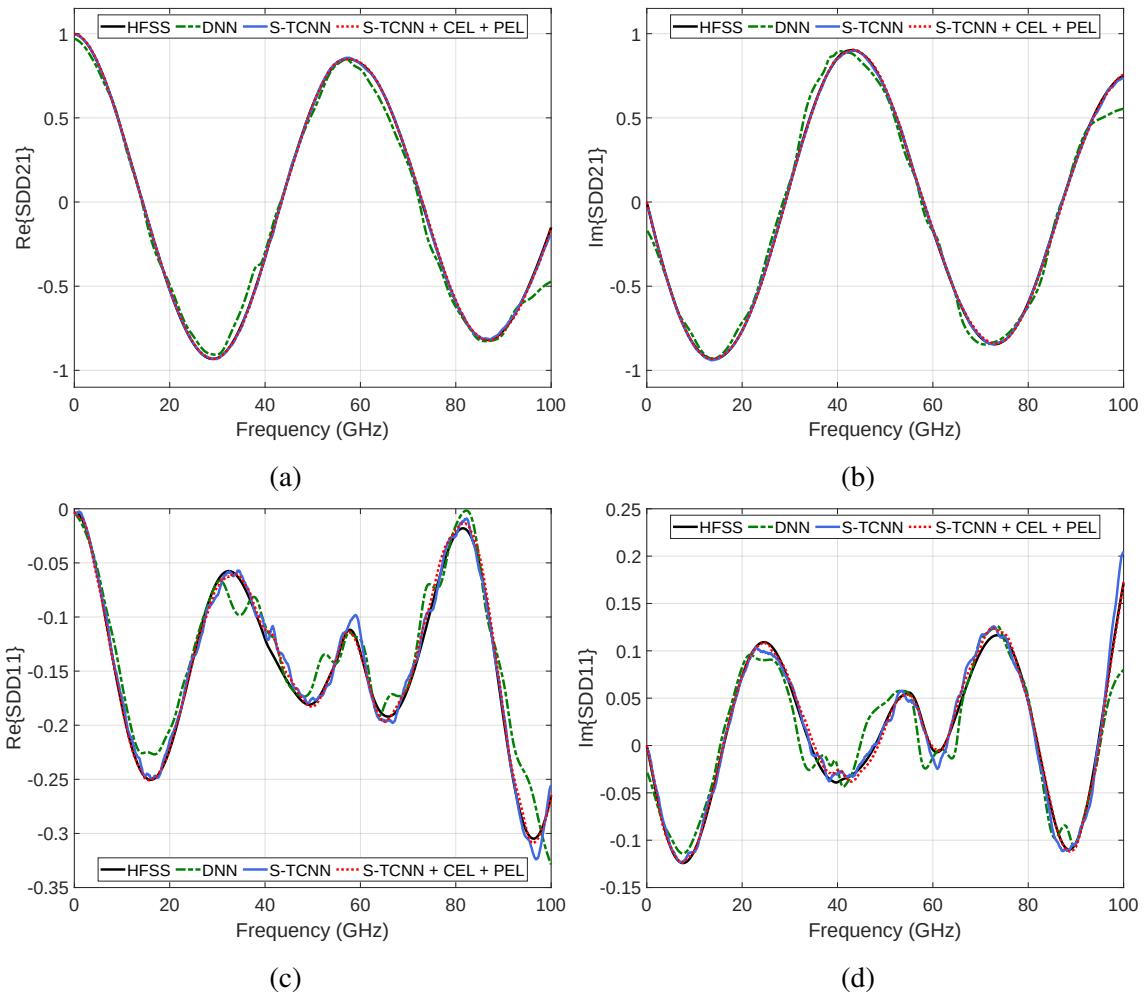


Figure 4.12: Comparison of predicted S-Parameters for different test cases of stripline model. (a,b) Real and imaginary part of differential IL. (c,d) Real and imaginary part of differential RL.

Table 4.4: Comparison of Models on Test Data for Stripline Model

	DNN	**S-TCNN**	**S-TCNN + CEL + PEL**
NMSE	4.73% $\pm$ 0.22%	3.08% $\pm$ 0.29%	2.47% $\pm$ 0.43%
Av. Causality Metric	8.24%	13.86%	100.0%
Range of σ_1	[0.818, 1.008]	[0.788, 1.004]	[0.812, 1.000]
Inference Time (for 1K broadband S-Params.)	0.11 s	0.09 s	0.61 s

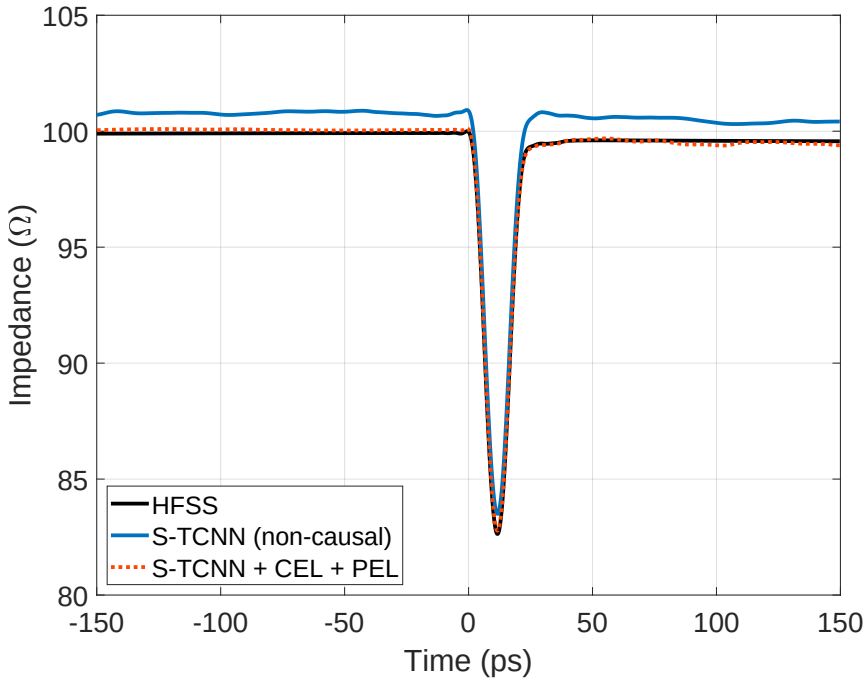


Figure 4.13: Comparison of differential TDR of predicted S-Parameters for the stripline model for a cosine-edge step with 15 ps 0-100% rise time.

around the resonances. Predicted S-Parameters using the S-TCNN and DNN model had an average causality quality metric of 13.86% and 8.24%, respectively, as compared to 100.0% with the proposed model. For the best performing two models, the effect of non-causality is further demonstrated in the differential TDR plot for a test case in Figure 4.13.

As the transmission line structure has a linearly increasing loss trend, maximum singular value of its S-Parameters linearly decrease as well. The high predictive quality of the models corresponds to capturing this linear trend very accurately. This results in almost passive S-Parameter predictions for S-TCNN and DNN models as the range of σ_1 is bounded by 1.004 and 1.008, respectively. However, the passivity is not guaranteed for further predictions that are not in training and test sets, whereas the proposed model guarantees this as shown by the range of σ_1 being bounded by 1.000 for the test set. The inference time for each model to generate 1000 broadband S-Parameter shows the minimal computational overhead property of the proposed layers.

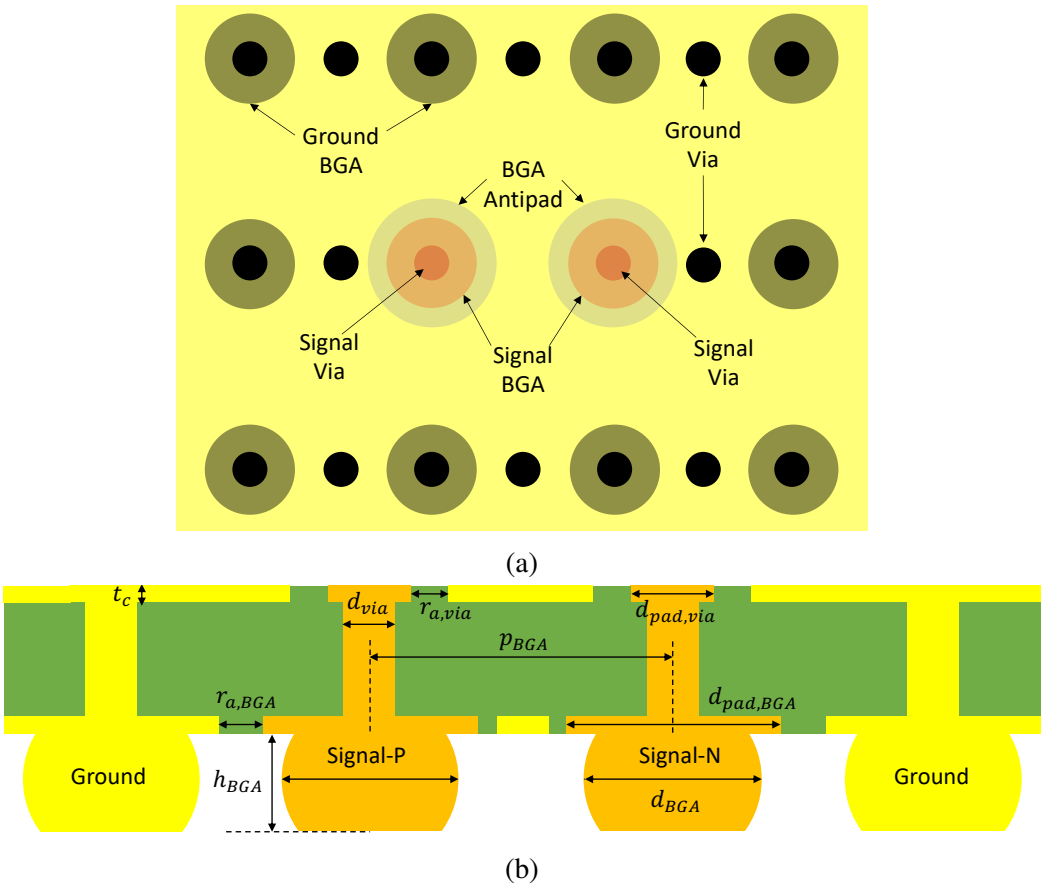


Figure 4.14: Geometry of the differential BGA structure in package-to-board transition. (a) Top view. (b) Cross-section.

4.5 Application 3: Differential BGA Pair and Cascade Analysis

The third application to demonstrate the effectiveness of the proposed model is a differential microvia to ball-grid-array (BGA) transition for package-to-board connections. These structures are the last step of the chip-to-board interconnects. Due to the relatively larger antipad diameters, such transitions can be highly capacitive and disrupt the signaling quality when operating at higher frequencies. As such, they need to be carefully designed to ensure desired bandwidth can be achieved. We therefore derive the parameterized model of the BGA structure given in Figure 4.14, where its geometry is parameterized using 9 variables within the bounds given in Table 4.5. The output parameters are the same as described in the PTH section, which represents 12,000 dimensions.

Table 4.5: Control Parameters of the BGA structure

Parameter		**Unit**	**Min**	**Max**
Via Diameter	d_{via}	μm	30	70
Via Pad Diameter	$d_{via,pad}$	μm	31	140
Via Antipad Radius	$r_{a,\ via}$	μm	50	500
BU Layer Thickness	h_{BU}	μm	20	35
Copper Thickness	t_c	μm	10	20
BGA Pitch	p_{BGA}	μm	300	1200
BGA Diameter	d_{BGA}	μm	180	960
BGA Height	h_{BGA}	μm	70	770
BGA Antipad Radius	$r_{a,BGA}$	μm	100	1050

Once the BGA model is verified, we then use it in cascade with the models for stripline and PTH to analyze the full vertical package-to-board transition in time-domain with the goal of demonstrating the significance of physical consistency of each block used in the cascade analysis.

4.5.1 Simulation Setup

For the BGA model, a total of 830 samples are collected and characterized between 0.1-100 GHz at 100 MHz steps using 3D EM solver. 700 of such simulations are used to train the predictive model and remaining 130 samples are used for validation. The ports are defined at antipads of microvias and bottom of the BGA, i.e. at the circular cross-section, where a perfect electric conductor (PEC) is used as the reference for the BGA ports.

4.5.2 Results

The comparison results are given in Table 4.6. Similar to the previous application examples, the DNN model had the worst predictive accuracy with an average NMSE of 4.74%, where convolutional type models performed significantly better with an average NMSE of 2.17% and 2.46% for S-TCNN and S-TCNN+CEL+PEL, respectively. A further comparison of actual and predicted S-Parameters are also given in Figure 4.15. The computational

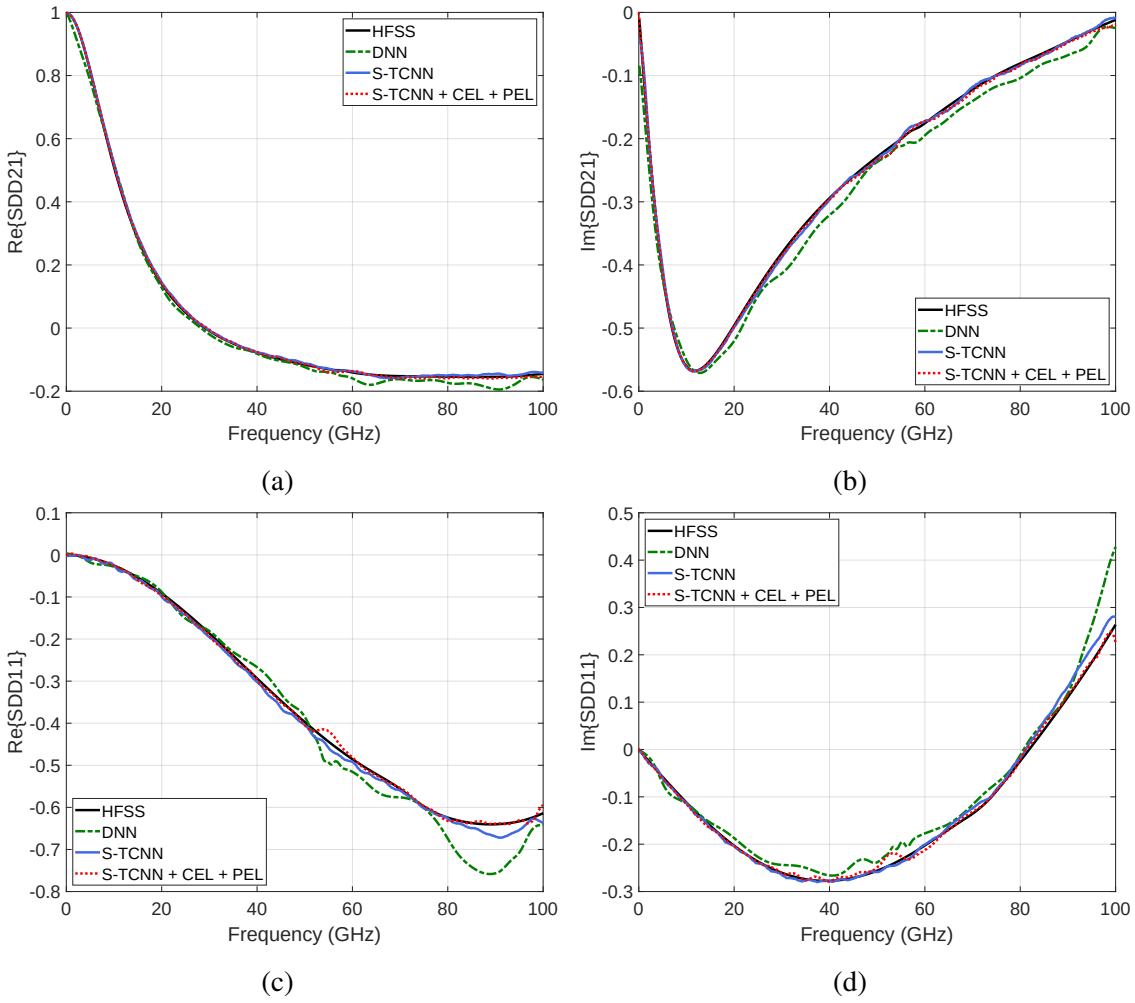


Figure 4.15: Comparison of predicted S-Parameters with 3D EM simulations for different test cases of BGA model. (a, b) Real and imaginary part of differential IL. (c, d) Real and imaginary part of differential RL.

Table 4.6: Comparison of Models on Test Data for BGA Model

	DNN	**S-TCNN**	**S-TCNN + CEL + PEL**
NMSE	4.74% $\pm$ 0.25%	2.17% $\pm$ 0.35%	2.46% $\pm$ 0.22 %
Av. Causality Metric	9.83%	13.71%	100.0%
Range of σ_1	[0.963, 1.277]	[0.962, 1.109]	[0.8225, 1.000]
Inference Time (for 1K broadband S-Params.)	0.18 s	0.01 s	0.57 s

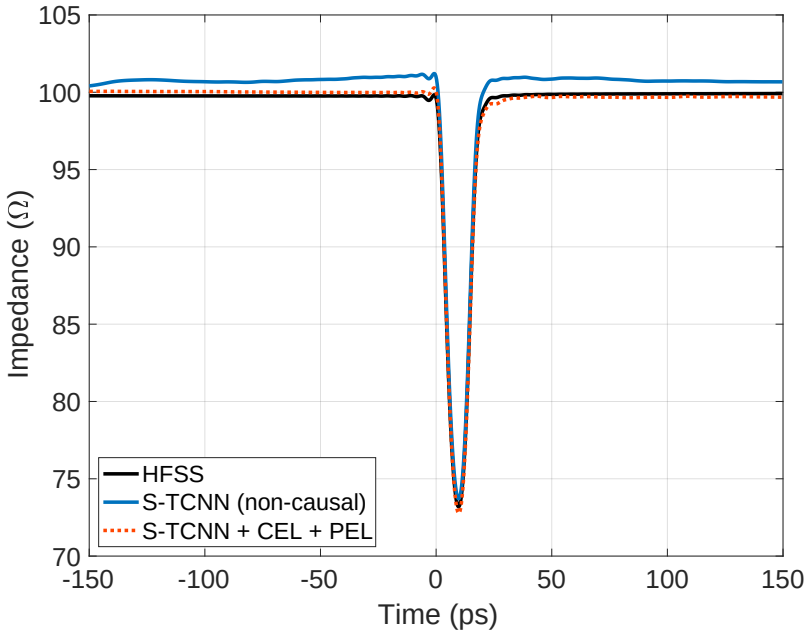


Figure 4.16: Comparison of differential TDR of predicted S-Parameters for the BGA model for a cosine-edge step with 15 ps 0-100% rise time.

overhead to inference time required to predict 1000 frequency responses are also observed to be minimal for the BGA model.

S-TCNN and DNN predicted S-Parameters had an average causality quality metric of 13.71% and 9.83%, and the maximum σ_1 was calculated to be 1.109 and 1.277, respectively, which shows significant causality and passivity violations. All the predicted S-Parameters for the proposed model were characterized as completely causal and passive. The effect of physical consistency is further demonstrated in Figure 4.16 via a differential TDR simulation.

4.5.3 Cascade Analysis

After the models for stripline, PTH and BGA are derived and their accuracy are verified, they can be cascaded together to obtain S-Parameter predictions for the full vertical package-to-board transition as in Figure 4.17. This allows for analyzing the TDR of the complete vertical transition that is required to perform a DSE and/or UQ to determine the most feasible design parameters to maintain a certain impedance along the transition. This

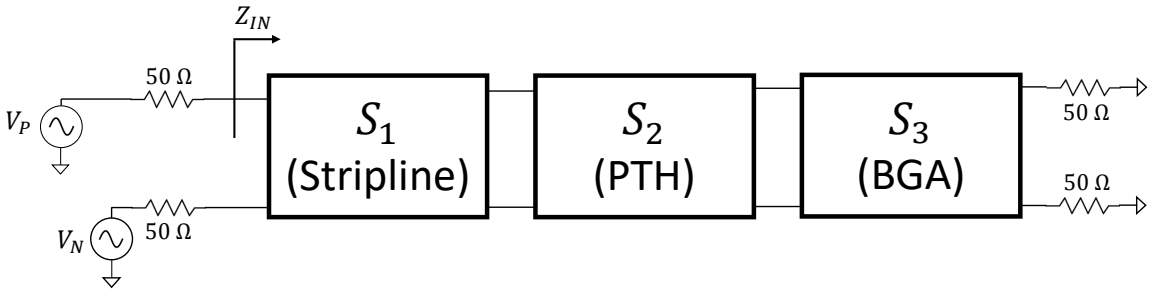


Figure 4.17: Schematic for analyzing the impedance of package-to-board transition by cascading predicted S-Parameters.

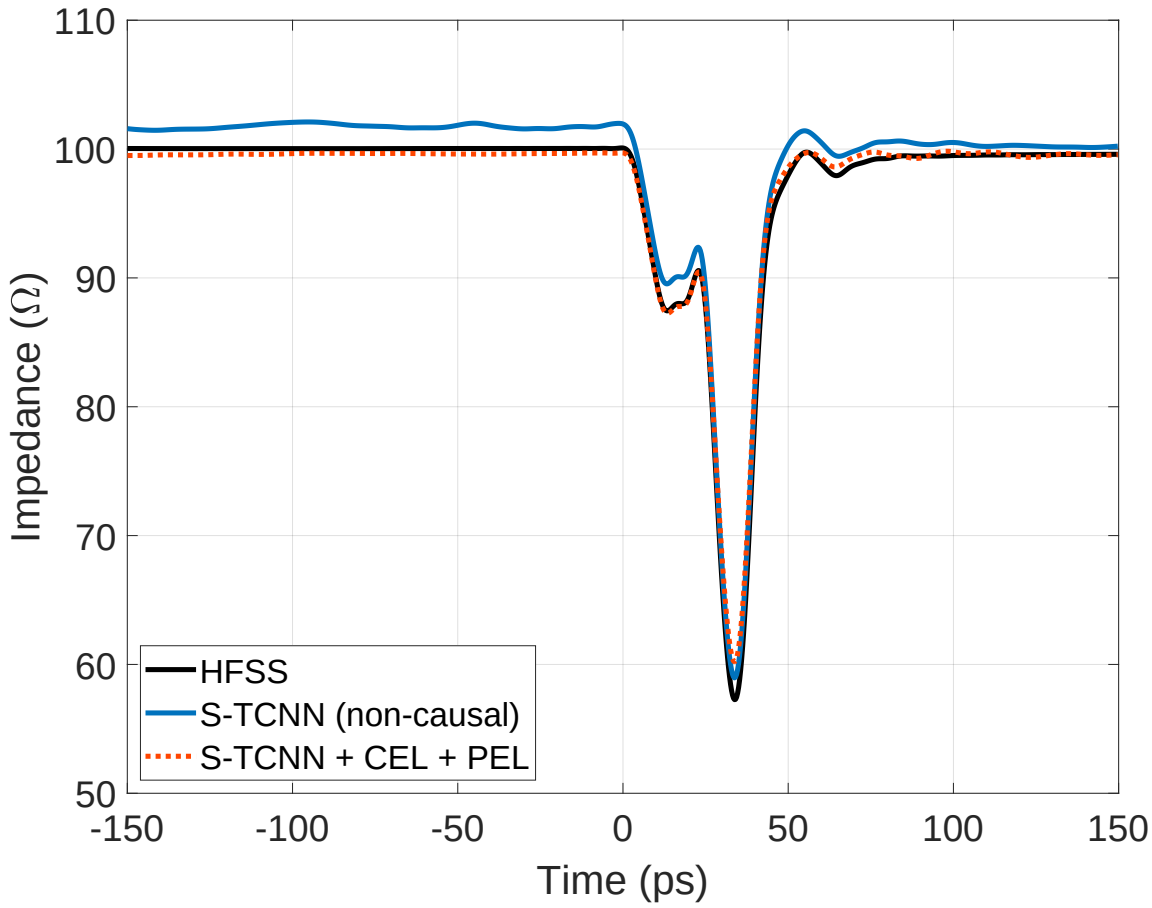


Figure 4.18: TDR comparison for cascaded analysis for a cosine-edge step input with 15 ps 0-100% rise time..

TDR analysis given in Figure 4.18 shows that as the proposed technique produces causal S-Parameter blocks, the resulting TDR from the cascade analysis is causal as well and has very good agreement with the full-wave simulation. The TDR obtained through S-Parameters predicted by S-TCNN model without CEL and PEL, on the other hand, shows noticeable causality violations.

4.6 Conclusion

In this chapter, we have shown that neural network (NN) models that are used to predict S-Parameters does not guarantee the physical consistency of predictions. Focusing purely on reducing numerical accuracy when creating the NN is therefore not suitable for predicting frequency responses. To this end, we have proposed *causality enforcement layer* (CEL) and *passivity enforcement layer (PEL)*, which ensures the predictions represent the frequency response of a causal and passive system.

In CEL, we have shown that truncation and discretization errors limit the direct use of Kramers-Kronig relations, and their effect can be minimized by NN based extrapolation and causality-preserving interpolation through exploiting properties of analytic signals. In PEL, we have proposed a new minimum-phase passivity enforcement filter to perform frequency-dependent passivity correction while not disrupting causality properties. Further, we have used an easily parallelized and computationally efficient upper bound to maximum singular value for the complex S-Parameter matrix to avoid costly SVD operations to minimize computational overhead.

We have demonstrated the effectiveness of our proposed model for 3 different application, where the goal was to parameterize 4-port single ended S-Parameter matrix from DC to 100 GHz, which correspond to 12,000 output dimensions. In all of the application, predictive accuracy of S-TCNN models with and without CEL and PEL were highly accurate when compared to 3D EM simulations and were very similar to each other. However, in the absence of PEL and CEL, the predicted S-Parameters showed significant causality

and passivity violations with causality quality metrics in the range of 11.17%-13.86% and maximum singular values in the range of 1.004-1.131 for the 3 applications. On the other hand, the proposed model have resulted in 100.0% causal and passive S-Parameters for all the test cases of all 3 application examples. The effect of physical consistency for each application is further demonstrated by using predicted S-Parameters in subsequent TDT and TDR simulations.

CHAPTER 5

BAYESIAN OPTIMIZATION FOR ELECTRONIC DESIGN AUTOMATION

Electronic design optimization is a widely studied area since the introduction of computeraided design (CAD) in 1970s that enabled electronic design automation (EDA). Since then, many techniques have been developed with the ultimate goal of designing a high performance and robust system. As the designs get more and more complicated due to continuously increasing scaling trend of ICs and passives to achieve higher overall performance while miniaturizing the system, it raises new challenges for developing new design optimization methodologies.

Often times, conventional techniques are not suitable for optimization of such design complexity due to many difficulties. Gradient-based methods, for instance, require approximating the gradient in a blackbox optimization setting, which can degrade the convergence rate and can quickly become computationally intractable. When the problem is non-convex, as common in the EDA domain, the algorithm needs to be initiated with random restarts to avoid getting stuck in many local optima points. Evolutionary algorithms, such as genetic algorithm and differential evolution, are commonly used alternatives. However, such techniques require a large number of generations and mutations to ensure convergence to the global optimum.

BO based on GPs is a widely used method in a variety of domains such as neuroengineering, aerospace engineering and material science and has recently received attention in the EDA domain [59, 60, 61, 62]. However, BO based algorithms utilized in prior work have been based on general purpose algorithms which do not specifically address the EDA related challenges such as increased number of control parameters and handling large sample spaces. Further, since the selection of acquisition function is a pre-determined strategy in general-purpose BO frameworks, there is no guarantee that a particular algorithm that

performed well on a particular design problem to behave the same at another.

In this chapter, we present a new BO based algorithm, Two-Stage Bayesian Optimization (TSBO), that is designed to address these challenges in the EDA domain. To handle large sample spaces, we eliminate the auxiliary optimization step from the BO framework by developing a new hierarchical partitioning tree based sampling strategy that allows to cover large design spaces of moderate dimensionality ($D \approx 10$) in an efficient manner. To make TSBO applicable to different designs problems across the electronics domain rather than being problem-specific, we present a sub-learning strategy that learns which acquisition function in the BO framework will work the best for the given problem. We demonstrate the performance of TSBO on four synthetic optimization test problems and two design application that emerge in packaging, namely thermal management for 3D ICs with a goal of skew minimization and 2) co-optimization of embedded inductors and integrated voltage regulators (IVR) to maximize voltage conversion efficiency.

5.1 Two-Stage Bayesian Optimization

In multi-armed bandit problems, the path to achieve optimal result goes through a trade off between exploration and exploitation as explained in Chapter 2. For the case of aforementioned acquisition functions, u_{PI} and u_{EI} in Equation 2.18 and Equation 2.19, the trade off is made with the introduction of the hyperparameter ζ. IMGPO handles the trade-off with the assumption of existence of a tighter bound than UCB [63] and BamSOO takes the approach of eliminating regions where with high probability, the region does not contain global optima [64].

In the new algorithm we present, TSBO, we approach this trade off in two stages, namely Fast Exploration Stage and Pure Exploitation Stage, which can also be interpreted as coarse and fine tuning respectively. In order to address aforementioned EDA related challenges, we present two additional techniques to increase convergence rate and extend the applicability of TSBO to a variety of electronics design problems. These consist of 1)

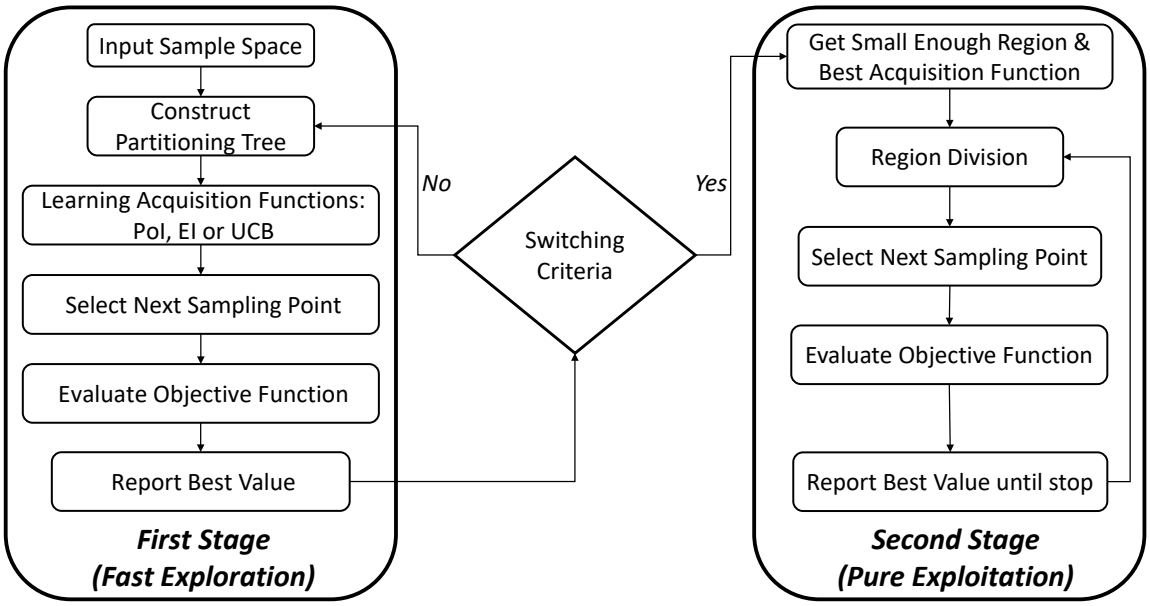


Figure 5.1: The flowchart of the TSBO algorithm.

learning acquisition functions and 2) a hierarchical partitioning tree construction scheme. The flowchart of TSBO is given in Figure 5.1.

5.2 Fast Exploration and Pure Exploitation Stages

The first stage of TSBO, *fast exploration stage*, can be considered as coarse tuning, as the purpose of this stage is to rapidly find the region, $\mathbb{A}^D$, in the sample space where the global optimum, $x^* = \arg\max_{x \in \mathbb{X}^D} f(x)$, is contained. For the purpose of finding a tight $\mathbb{A}^D$ rapidly, at t^{th} iteration of optimization, we divide the sample space $\mathbb{X}^D$ into 2^D regions of hyper rectangles, H_t, and generate candidate points, $c_{i,j}$, to determine the next sampling point, x_{t+1}. These candidate points are chosen as the center of each region, given as

$$c_{t,j} = \frac{H_{t,j,min} + H_{t,j,max}}{2}, \quad j = 1, 2, ..., t2^D \tag{5.1}$$

where $c_{t,j}$ is the j^{th} candidate point of a total of $t2^D$ points at iteration t; $H_{t,j}$ is the region that $c_{t,j}$ belongs to and $H_{t,j,min}$ and $H_{t,j,max}$ are corresponding lower and upper boundaries of each D dimensional region. In order to avoid relying on auxiliary optimization to select

x_{t+1}, the acquisition function, $u(x)$, is not optimized but evaluated at each candidate point as follows

$$c_{(t,j^*)} = \arg\max_{c \in C} u^{(i)}(c) \tag{5.2}$$

$$x_{t+1} = c_{(t,j^*)} \tag{5.3}$$

where j^* denotes the selected candidate; C represents the finite set of candidate points and $u^{(i)}(x)$ is u_{UCB}, u_{EI} or u_{PI}, selected sequentially as explained later in the section on *learning acquisition functions*. After querying the function at x_{t+1}, the new set of regions are generated using

$$H_n = \bigcup_{i=1}^{2^d} h_i, \quad h_i \subset H_{(t,j^*)} \tag{5.4}$$

$$H_{t+1} = H_n \cup H_t \tag{5.5}$$

where H_n is the union of new regions, h_i, acquired by dividing H_{t,j^*} into 2^d hyper rectangles.

The first stage remains until it explores a small enough region ($x^* \in \mathbb{A}^d$) such that the euclidean distance between *previous best* and *current best* input points are negligible. This is illustrated in Figure 5.1 as the switching criteria, given as

$$n = \begin{cases} n+1, & \text{if} \quad \|x_{max} - x_{pmax}\| < 10^{-3} \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

where x_{max} and x_{pmax} are the input vector of current and previous best observations re-

spectively and switching occurs if $n > N$. The selection of $\mathbb{A}^d$ can be written as

$$\mathbb{A}^d = \begin{bmatrix} (1-\alpha)x_{max,1} & (1+\alpha)x_{max,1} \\ \vdots & \vdots \\ (1-\alpha)x_{max,d} & (1+\alpha)x_{max,d} \end{bmatrix} \tag{5.7}$$

where $x_{max,i}$ is the i^{th} dimension of x_{max} in Equation 5.6 and α is a hyper parameter of TSBO for choosing the tightness of the region provided to the second stage.

The second stage of TSBO, namely *pure exploitation stage*, takes $\mathbb{A}^d$ and $u^*(x)$ from the first stage as inputs and performs fine tuning for the optimization problem, i.e. increases accuracy. At this stage, the tight region $\mathbb{A}^d$ is divided into 3 new regions along its longest dimension at each iteration and candidate points are generated at each region in the same fashion as the first stage, but using the learned acquisition function of $u^*(x)$.

5.2.1 Hierarchical Partitioning Tree

TSBO uses a distinctive hierarchical partitioning tree that differentiates it from general purpose BO algorithms and makes it more EDA oriented. Typically, a hierarchical partitioning tree is constructed via fully committing to and expanding the selected branch, i.e. sampling each child node [65]. For example, BamSOO uses a region shrinking technique and the tree is expanded only in non-discarded region [64]. However, the child node of the selected branch is fully expanded if it belongs to a non-discarded region.

In TSBO, the selected branch is expanded to generate 2^D candidate points, i.e. child nodes, but the sampling only occurs at the most promising child node, hence, only one function query occurs per iteration. This overcomes the limitation in the number of branches that can be generated as in BamSOO and IMGPO and allows for rapid coverage of the sample space. In other words, with the cost of a few seconds of algorithm run time due to storing $t2^D$ points at each iteration, TSBO substantially reduces the number of required simulations needed to find x^*. An example partitioning tree constructed using TSBO and

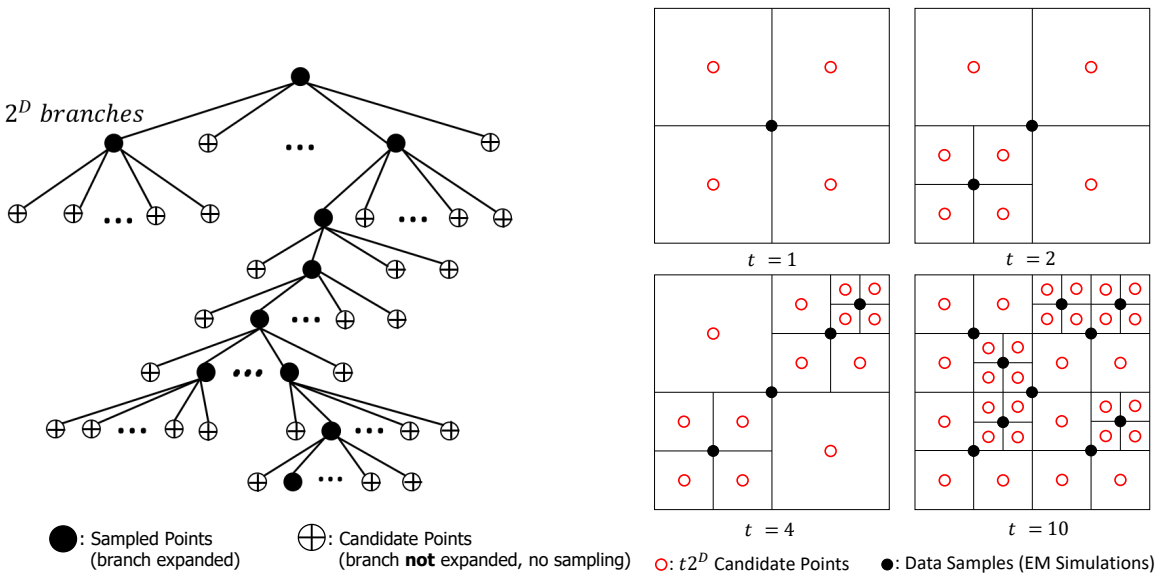


Figure 5.2: Partitioning tree strategy to cover large sample spaces. (a) Topological view of an example tree structure. (b) Sample space view for a 2-D example.

its progression for a 2D sample space is given in Figure 5.2.

5.3 Learning Acquisition Functions

As the acquisition function is a pre-determined strategy, there is no guarantee that a single strategy will outperform others for every problem as each has their advantages [66]. For instance, PI in Equation 2.18 is biased towards exploitation over exploration. This limits its use for many problems, but very advantageous when the objective function has a narrow valley, such as encountered in microwave filters [67]. EI in Equation 2.19 can better handle the exploration and exploitation trade-off in larger sample spaces. UCB in Equation 2.20, on the other hand, is better for optimization with a limited budget due to employing a multi-armed bandit formulation that tries to maximize overall gain under the budget. Since the limited budget setting holds for many practical design optimization problem, UCB becomes an attractive option.

In TSBO, we pose selecting the right acquisition function problem as a *sub-learning task* in the overall optimization loop. For the first K iterations of the optimization, we se-

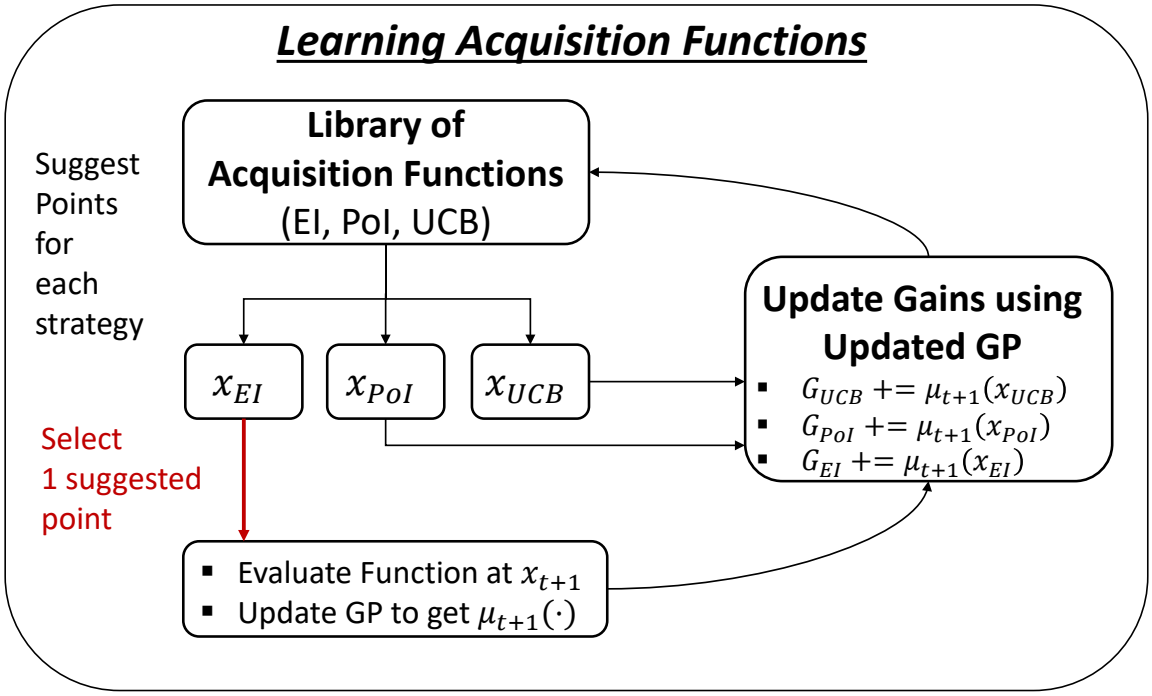
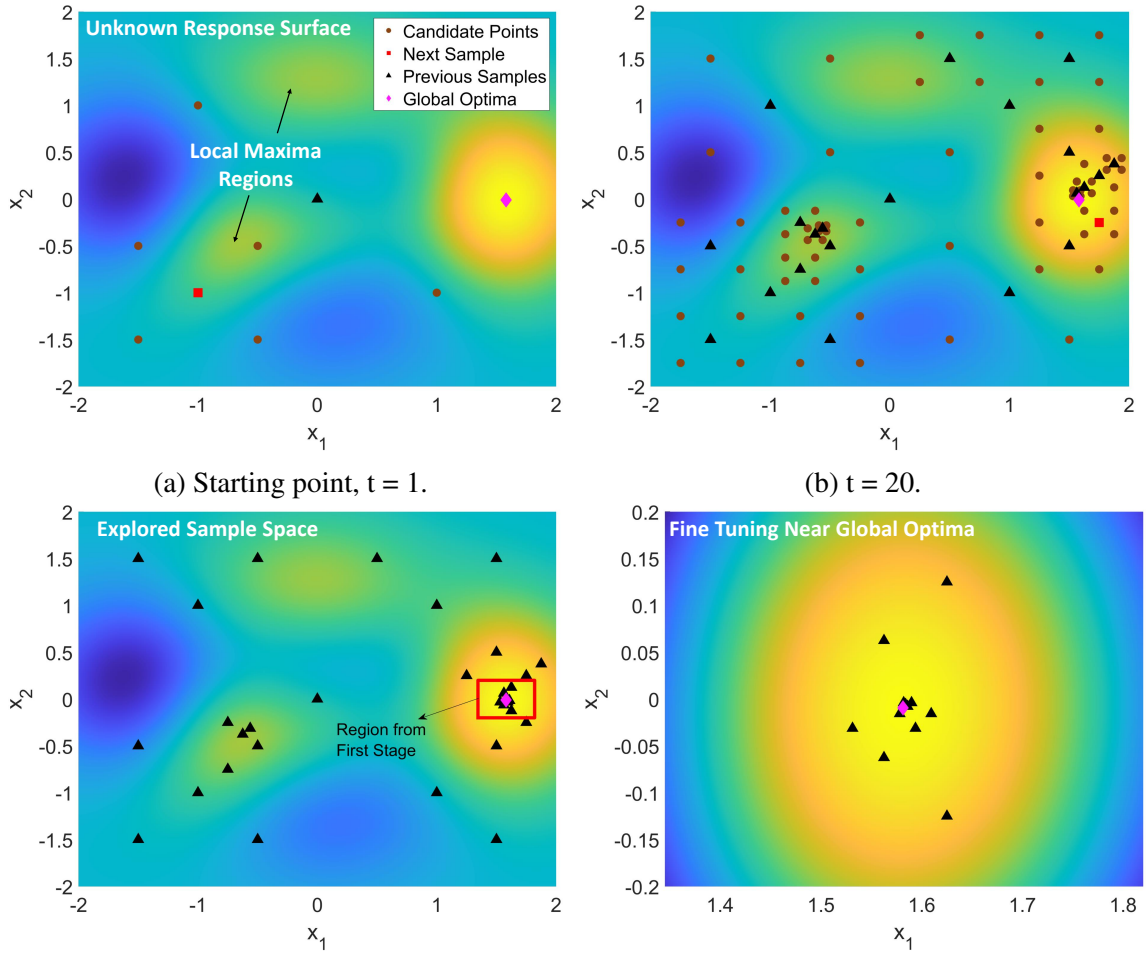


Figure 5.3: Learning acquisition functions in TSBO.

quentially select either PI, EI or UCB to determine x_{t+1}, but also store the points suggested by each strategy as x_{UCB}, x_{EI} and x_{PI}. Once the objective function is evaluated at x_{t+1} and the GP model is updated, we predict what would have happened if we had simulated at other suggested points. To do this, we evaluate the points suggested by other strategies using the posterior mean of the updated GP, $\mu_{t+1}(\cdot)$. The predictions from the updated GP is added cumulatively to get a "gain" for each strategy, and after K iterations, we select the one with the highest gain, denoted as $u^*(x)$. This process allows to benefit from each strategy for the first K iterations and then continue with the best for the remainder of the optimization. Hence, it makes TSBO applicable to various design optimization problems with varying response surface structures rather than being specific to a single problem type. This process of sub-learning the acquisition functions is summarized in Figure 5.3.



(a) Starting point, t = 1.

(b) t = 20.

(c) End of first stage, t = 42.

(d) End of second stage, t = 100.

Figure 5.4: Progression of TSBO for optimizing the 2D Peaks function.

5.3.1 Experiments on Synthetic Benchmark Functions

As an example to demonstrate how TSBO works, we apply it to maximize the 2D *peaks* function that is available in MATLAB. As shown in Figure 5.4a, the optimization starts with only 1 data point. As the algorithm progresses into 20 iterations as in Figure 5.4b, candidate points start to cover the entire sample space, but the active learning strategy directs function evaluations to concentrate at interesting regions, i.e. near local and global maximum. The first stage automatically ends after 42 iterations and the small region (red rectangle in Figure 5.4c) is passed to the second stage to perform fine tuning until a pre-determined number of simulations is completed as in Figure 5.4d. An open-source MATLAB implementation

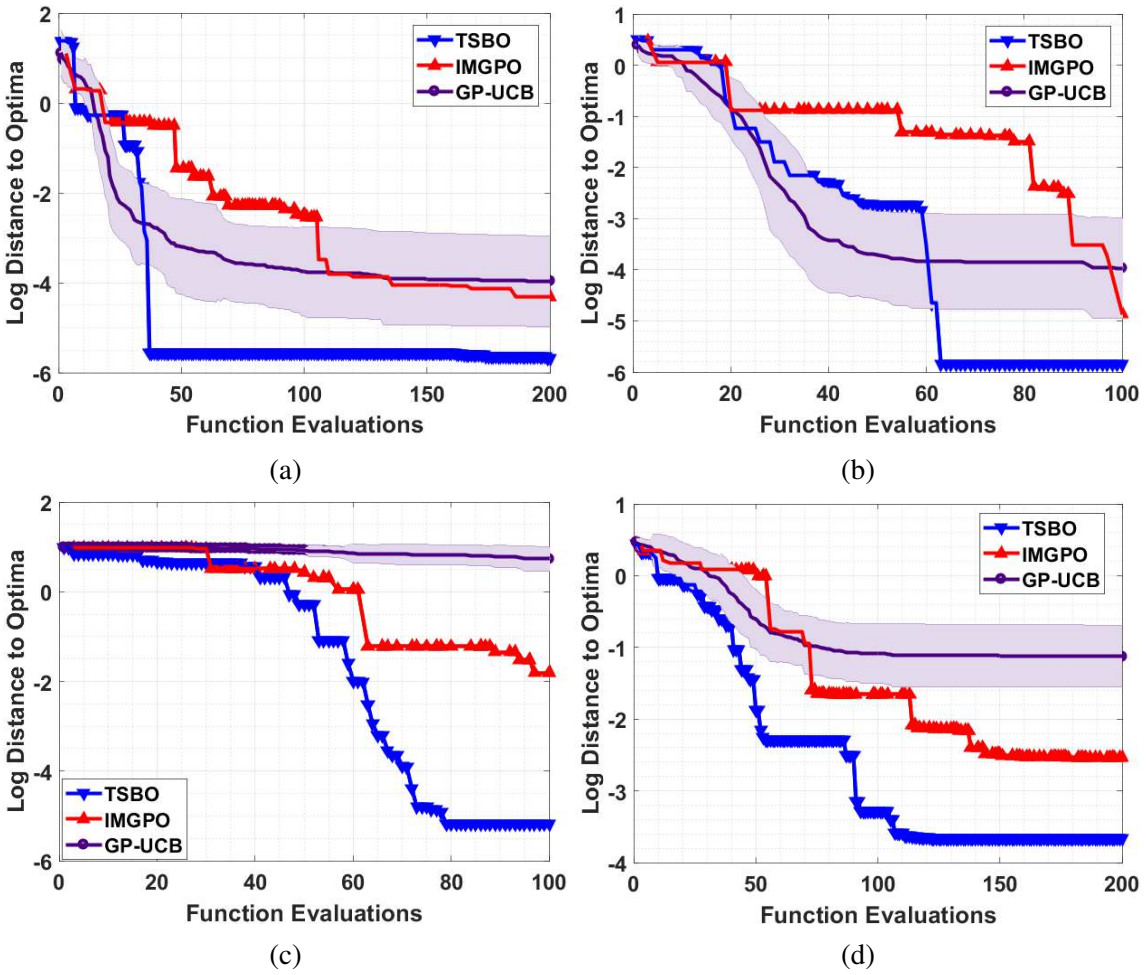


Figure 5.5: Performance comparison for proposed algorithm, TSBO, on optimization synthetic functions used in literature. Functions are available in [68]. (a) 2-D Branin. (b) 3-D Hartmann. (c) 4-D Shekel5. (d) 6-D Hartmann

of TSBO can be found at `https://github.com/GT-PRC/TSBO`.

In order to test the performance of TSBO, we have considered four different synthetic functions with different dimensions, which are common benchmarking problems in optimization [68]. Performance evaluation criteria in these experiments shown in Figure 5.5 is *log distance to optima*, i.e. $log_{10}(|f^* - \widetilde{f}^*|)$. Kawaguchi showed that IMGPO outperforms BamSOO, GP-EI and GP-PI on the same synthetic functions used in this work [63], hence, we compare TSBO with IMGPO and GP-UCB using MATLAB's *patternsearch* for auxiliary optimization of GP-UCB. Since the initial point is selected randomly in GP-UCB, we repeat the experiments 50 times and report the mean and standard deviation. As shown

Table 5.1: Algorithm Progression Times and Accuracy within 100 Iterations

	GP-UCB		**IMGPO**		**TSBO**	
	Run Time (Var. Memory)	Accuracy	Run Time (Var. Memory)	Accuracy	Run Time (Var. Memory)	Accuracy
2D Branin	10.38s (0.99 Mb)	$10^{-3.8}$	1.28s (1.65 Mb)	$10^{-2.5}$	6.15s (2.85 Mb)	$10^{-5.6}$
3D Hartmann	10.52s (0.10 Mb)	10^{-4}	1.35s (1.66 Mb)	$10^{-4.8}$	4.35s (2.01 Mb)	$10^{-5.8}$
4D Shekel5	11.35s (0.10 Mb)	$10^{0.8}$	1.83s (1.68 Mb)	$10^{-1.8}$	6.79s (3.17 Mb)	$10^{-5.2}$
6D Hartmann	12.23s (0.11 Mb)	$10^{-1.2}$	1.91s (1.70 Mb)	$10^{-2.5}$	6.51s (2.19 Mb)	$10^{-3.3}$

in Figure 5.5, TSBO outperforms IMGPO and GP-UCB in these benchmark problems in terms of requiring less function evaluations for converging to the global optimum.

Furthermore, algorithm progression times (excluding function evaluation time) and total variable memory stored for 100 iterations are measured and compared in Table 5.1. Although TSBO achieves higher accuracy with less function evaluations, it requires additional run time and memory to store and process the aforementioned $t2^D$ candidate points at each iteration. As IMGPO uses a partitioning scheme that has lighter weight on memory compared to TSBO, it requires lesser processing at each iteration which results in faster run times. GP-UCB uses the least amount of memory among the algorithms considered as it does not store additional variables at each iteration, however, its run time is the highest due to the auxiliary optimization process.

In addition, run time statistics summarizing the effect of learning acquisition functions block of TSBO are provided in Table 5.2. Here, number of calls refer to number of times that particular $u(x)$ has been used to select x_{t+1}. It can be seen that each $u(x)$ contributes to the optimization process to find x^*, however, the amount of information gained from each $u(x)$ is different for each problem. As a result, the acquisition function providing

Table 5.2: Run Time Statistics of Learning Acquisition Functions block of TSBO for Synthetic Functions

	PI		**EI**		**UCB**	
	Normalized Gain	Number of Calls	Normalized Gain	Number of Calls	Normalized Gain	Number of Calls
2D Branin	0.2669	17	0.3164	16	0.4167	67
3D Hartmann	0.3205	16	0.3475	68	0.3321	16
4D Shekel5	0.4074	22	0.4228	62	0.1698	16
6D Hartmann	0.3352	16	0.3461	68	0.3207	16

the highest gain is called more by TSBO. Although the gains for each $u(x)$ are being dynamically updated after M observations as explained in previous section, the choice of $u^*(x)$ at M^{th} observation is only updated for the case of 4D-Shekel5 function as more observations are added.

The choice of hyperparameters can significantly affect the performance of blackbox optimization algorithms, including TSBO. Here, problem specific knowledge can be leveraged to adjust these parameters. For instance, if the problem is believed to have large number of local extrema, the algorithm can be adjusted to give more importance to exploration than exploitation. For the case of BO based algorithms, the effect of hyperparameters is a well-studied subject, resulting in rule of thumb values [69, 70, 71]. Counter-intuitively, these studies suggest dynamic modulation of these parameters have little to no-effect on empirical performance of the algorithms. Following these works, for these experiments, the hyperparameters of TSBO are chosen as: $\alpha = 0.1$ in Equation 5.7; $\eta = 0.1$ in Equation 2.20; $\zeta = 0$ and $\zeta = 0.01$ in Equation 2.18 and Equation 2.19, respectively; $M = 50$ for learning acquisition functions and $N = 10$ for switching criteria. TSBO specific parameters such as N, M and α can be modified accordingly to transfer domain expertise to

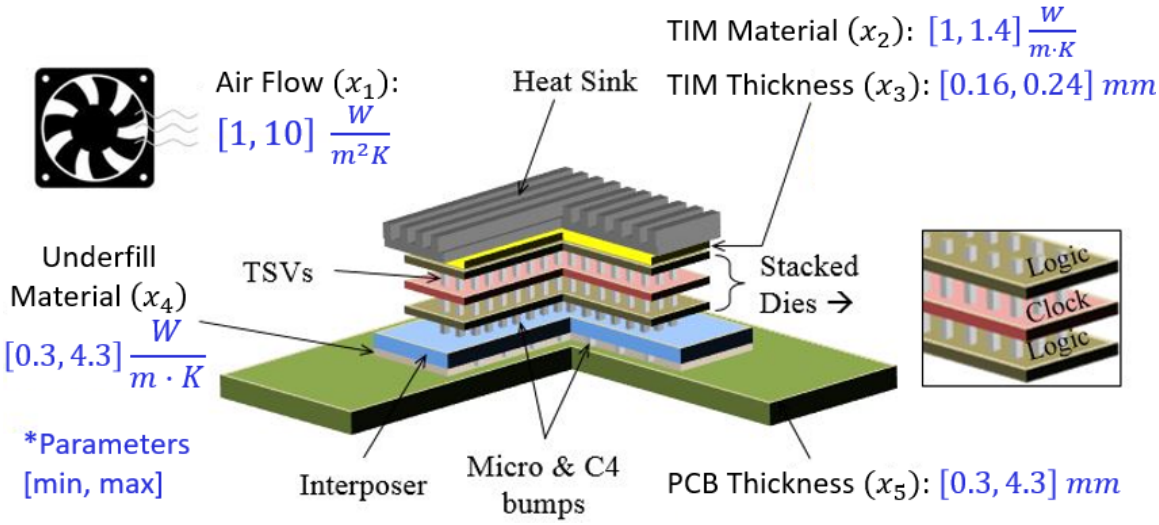


Figure 5.6: Geometry of the stacked 3D IC.

the algorithm. For instance, if the design under consideration is to be fabricated using a process that can not provide accuracy beyond microns, fine tuning beyond this level using second stage of TSBO may result in invalid designs. Hence, N can be increased to force TSBO to spend more time in the first stage.

5.4 Application 1: Clock Skew Minimization for 3D ICs

The first application chosen to evaluate TSBO is thermal management of 3D ICs, in particular, to minimize the global skew caused by temperature and temperature gradients in 3D ICs. 3D integration results in increased temperature and temperature gradient that degrades signal integrity. Furthermore, the multi-scale geometry of the 3D ICs requires CPU extensive simulations as finer mesh grids are used to capture variations in stacked dies connected with through silicon vias (TSV). The geometry of the 3D integrated system comprising of stacked dies, interposer, and printed circuit board (PCB) is given in Figure 5.6

Previous work has presented and verified an accurate modeling and simulation platform that uses an electrical-thermal solver [72] to calculate thermal characteristics and used BO to control temperature and temperature gradients to minimize clock skew [73]. In this work, we use the same problem of clock skew minimization for 3D ICs in [73]

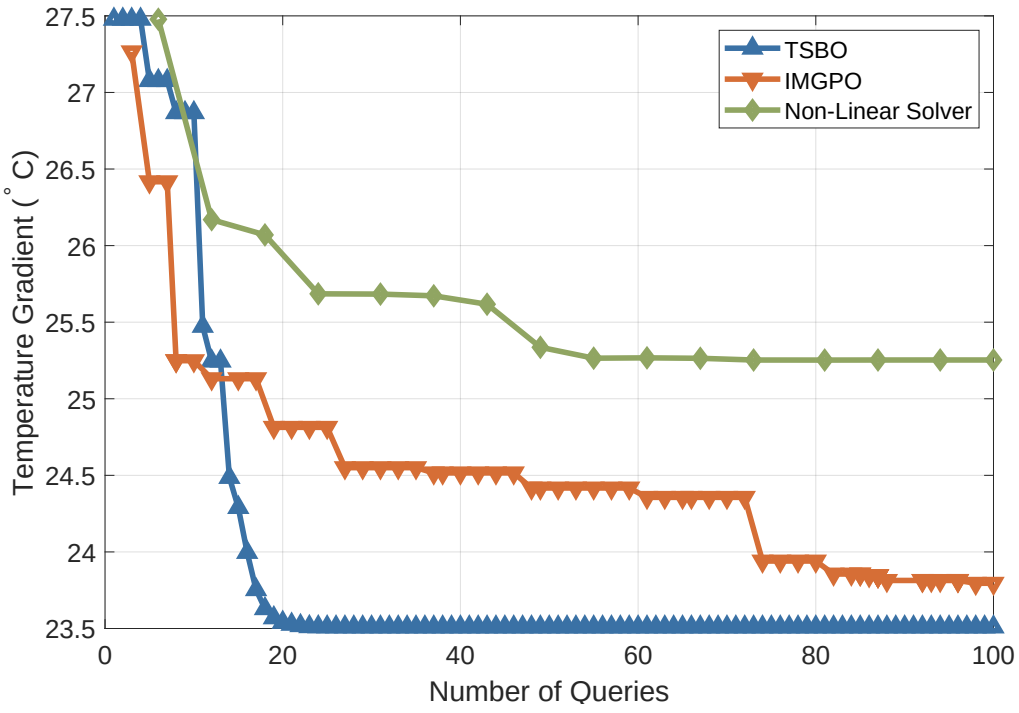
Table 5.3: Optimization Results for 3D IC

	Non-Linear Solver	**Previous Work [73]**	**This Work**
T_{GRAD} [°C]	25.2 (+9.4%)	23.8 (+4.7%)	23.5
Skew [ps]	96.6 (+12.3%)	88.0 (+2.3%)	86.0
CPU Time (Normalized)	3.96	3.76	1.00

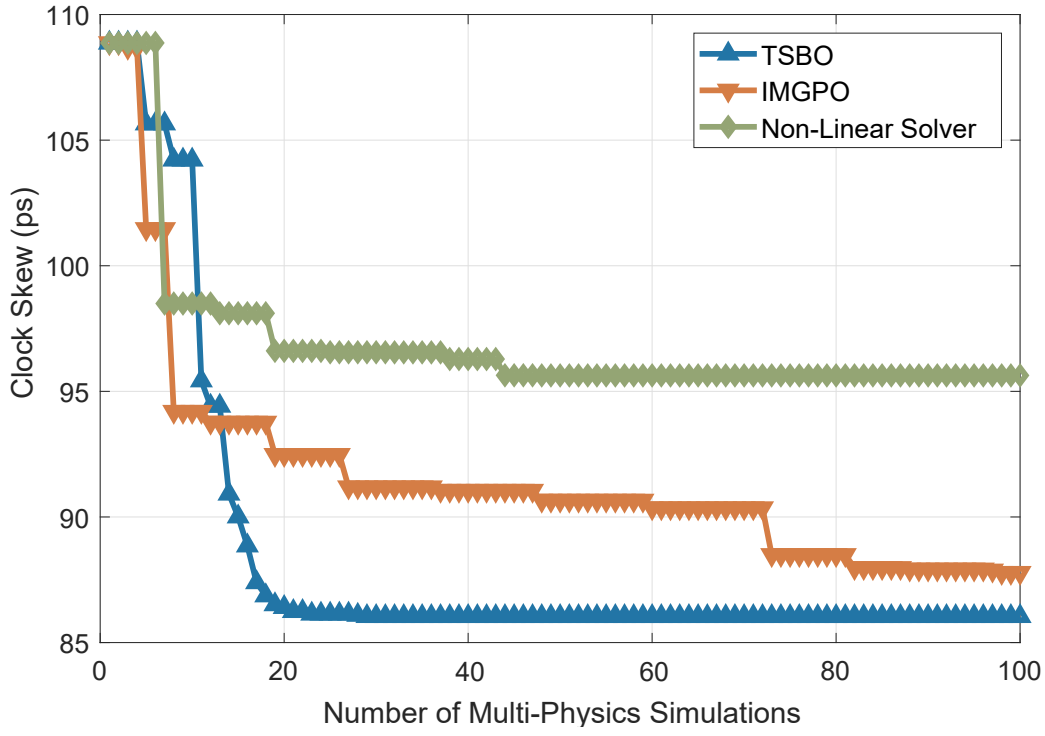
by controlling temperature profile to show applicability and convergence trend of TSBO. The problem is posed as a 5 dimensional problem which includes the control of heat transfer coefficient, Thermal Interface Material (TIM), TIM thickness, Underfill material and printed circuit board material with corresponding bounds shown in Figure 5.6. For brevity, we refer readers to [73] for the detailed description of the problem along with the accuracy of the simulation framework as compared to measurement results.

Simulations used in this work assumes a random generated static power map on the chip, as in [73]. Temperature profiles are calculated using aforementioned electrical-thermal solver, which uses finite volume method to simultaneously solve voltage distribution and thermal equations considering temperature dependent electrical resistivity and Joule heating effect[72]. The solver uses 3D nonuniform mesh and domain decomposition to capture the multi-scale geometry of 3D IC. Calculated temperature profiles are then fed into commercial circuit solvers to perform temperature dependent electrical simulation for signal integrity analysis, which provides the clock skew. The optimization objective is chosen as the minimization of temperature gradient which results in minimized clock skew.

The results of the optimization are summarized in Table 5.3 and Figure 5.7. Convergence curves in Figure 5.7 show the change in temperature gradient and clock skew with function queries, i.e. number of simulations. Previous work has shown that IMGPO and non-linear solver outperforms other non-ML algorithms including pattern search, genetic algorithm and multistart method[73]. To make a direct comparison, we use two best algorithms considered in [73] and compare performance of TSBO with non-linear solver



(a)



(b)

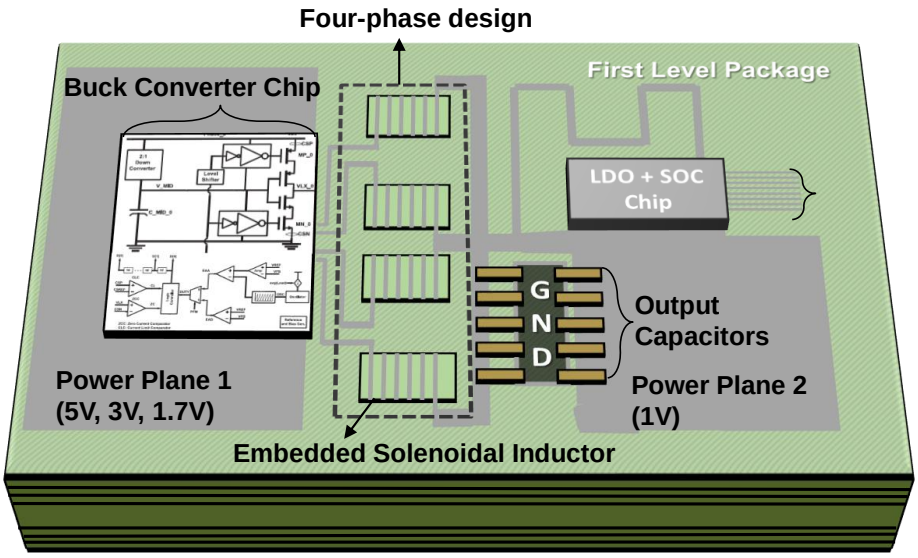
Figure 5.7: Performance of the TSBO algorithm for clock skew minimization.

(*fmincon* in MATLAB) and IMGPO. Optimization using TSBO resulted in a temperature gradient of 23.5 °C and clock skew of 86.0 ps, compared to 23.8 °C and 88.0 ps presented as preceding work with IMGPO and 25.2 °C and 96.6 ps with non-linear solver, which corresponds to an improvement of 4.7% and 2.3% as compared to IMGPO and 9.4% and 12.3% as compared to non-linear solver. The most compelling contribution of the algorithm is the significant decrease in CPU time required for convergence where TSBO converged to minimum temperature gradient 3.76 and 3.96 times faster than non-linear solver and previous work, respectively. The results of optimization for the 3 different algorithms are provided in Table 5.3. Note that the normalized CPU times provided are calculated as the combined time of system simulations and algorithm progression time to converge to their corresponding minimum skews.

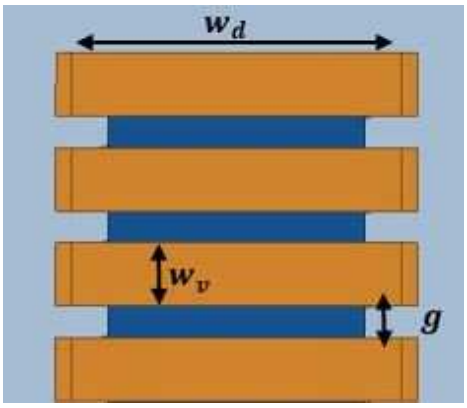
5.5 Application 2: Maximizing Efficiency in IVRs

The second application chosen in this work is the multi-objective optimization of IVR with the goal of maximizing power efficiency and minimizing area of embedded inductor. The architecture used is given in Figure 5.8a and is a System-in-Package (SiP) solution consisting of two chips (buck converter and LDO/load) with integrated inductor on an organic package [74]. The inductor structure is chosen as a solenoid with magnetic core as in Figure 5.8b and Figure 5.8c. Due to high levels of integration, maintaining high conversion efficiency while minimizing the area of inductor in the package becomes a major challenge that requires the handling of multiple trade-offs simultaneously.

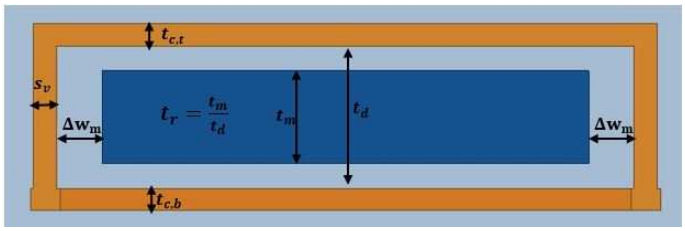
In this section, we use TSBO to automate the design process. We briefly provide previously developed buck converter efficiency model followed by the embedded inductor characterization using two commercial EM solvers, namely Ansys HFSS and Ansys Maxwell for full-wave simulations at high frequency and at DC, respectively. Afterwards, we provide the optimization setup used and make a comparison of the optimized IVR design with hand tuned design [74] as well as results generated using other non-ML and ML techniques



(a)



(b)



(c)

Figure 5.8: Four-phase SiP Integrated Voltage Regulator (IVR). (a) Overall IVR solution. (b) Top view of inductor. (b) Side view of inductor.

such as non-linear solver, GP-UCB and IMGPO.

5.5.1 Buck Converter Efficiency Model

The buck converter shown in Figure 5.8a is designed with stacked topology, using 130nm GF process and consists of four-phases (one master-three slaves) [74]. Efficiency calculations are based on the extensive model that accounts for switching and conduction losses of power switches, DC and AC losses of inductor, power delivery network (PDN) and output capacitance. Power switches used in the buck converter introduces two types of losses,

namely switching and conduction. Conduction loss originates from the finite DC resistance of switches and can be written as

$$L_{PS(COND)} = R_{SW}(I_{LOAD}^2 + \Delta I_{RMS}^2) \tag{5.8}$$

$$R_{SW} = D_{CCM}R_{PMOS} + (1 - D_{CCM})R_{NMOS} \tag{5.9}$$

where R_{NMOS} and R_{PMOS} are ON resistances of NMOS and PMOS respectively and ΔI_{RMS} is the ***RMS*** value of ripple current calculated as

$$D_{CCM} = V_{OUT}/V_{IN} \tag{5.10}$$

$$\Delta I_{pk-pk} = \frac{(V_{IN} - V_{OUT})D_{CCM}}{f_{SW}L} \tag{5.11}$$

$$\Delta I_{RMS} = \Delta I_{pk-pk}/2\sqrt{3} \tag{5.12}$$

where D_{CCM} is duty cycle in continuous conduction mode. The switching loss of power switches is given as

$$\begin{aligned} L_{PS(SW)} = & f_{SW}\big[(C_{GSN} + C_{GSP})(V_{IN}/2)^2 ... \\ & ... + 2(C_{GDN} + C_{GDP})(V_{IN}/2)^2\big] \end{aligned} \tag{5.13}$$

where C_{GSP}, C_{GSN}, C_{GDP} and C_{GDN} are gate-to-source and gate-to-drain capacitances of PMOS and NMOS respectively. The contribution of the inductor in terms of AC and DC resistances can be written as

$$L_{IND,DC} = I_{LOAD}^2 R_{IND,DC} \tag{5.14}$$

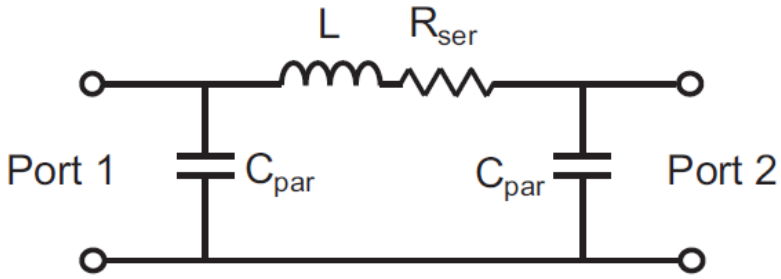


Figure 5.9: π-equivalent inductor model

$$L_{IND,AC} = \sum_{f=-\infty}^{\infty} 2\text{x}\frac{\Delta I_{pk-pk}^2}{4}\left[0.405^2\text{x}ESR_L(f_{SW})... \right. \\ \left. ... + 0.045^2\text{x}ESR_L(3f_{SW}) + 0.016^2\text{x}ESR_L(5f_{SW})\right] \tag{5.15}$$

where $R_{IND,DC}$ and ESR_L are DC and frequency dependent effective series resistance of inductor respectively; 0.405, 0.045 and 0.016 are Fourier series coefficients of fundamental frequency and its third and fifth harmonics of triangular current waveform. Resistive losses due to output capacitor and PDN are given as

$$L_{CAP} = \Delta I_{pk-pk}^2 ESR_C \tag{5.16}$$

$$L_{PDN} = I_{LOAD}^2 R_{PDN} \tag{5.17}$$

where R_{PDN} is effective DC resistance of PDN and ESR_C is effective series resistance of output capacitance. A more detailed description of the buck converter topology along with the model verification can be found in [74].

5.5.2 Embedded Inductor Characterization

Although there are approximate closed form expressions for preliminary design of embedded solenoidal inductors with magnetic core [75], a full-wave simulation is required to accurately account for eddy currents, proximity and skin effect as well as demagnetization effect due to using magnetic cores. Electrical characteristics of inductor have both direct and implied effects to the efficiency of the buck converter. The inductance needs to be

sufficiently high in order to reduce the ripple current in Equation 5.11, thereby reducing the conduction loss in Equation 5.9. The ripple current also affects the AC loss in inductor along with ESR in Equation 5.15. Although DC resistances of such inductors are in the range of miliohms, operating at higher currents ($\sim$10A) introduces a substantial DC loss as in Equation 5.14. Therefore, the complexity of the problem includes handling inductance, AC and DC resistance trade-offs and area constraints, while considering the direct and implied effects on buck converter efficiency. Even if the desired characteristics of the inductor is determined to handle these trade-offs, determining the dimensions of the inductor that will satisfy these characteristics requires substantial human intervention and CPU time due to the use of full-wave EM solvers. In order to characterize the inductor behavior, the full-wave EM simulation is conducted to gather 2-port impedance matrix as a function of frequency. Using a pi-equivalent circuit as in Figure 5.9, the inductance and ESR are calculated from the Z-matrix as

$$L = \frac{2Im\{Z_{11} - Z_{12}\}}{w}, \qquad ESR = 2Re\{Z_{11} - Z_{12}\} \tag{5.18}$$

Finally, Ansys Maxwell is used to accurately simulate the DC resistance of inductor to be used in Equation 5.14 of the efficiency model. In this work, we consider two different magnetic materials, namely Carbonyl Iron Powder (CIP) and Nickel-Zinc (NiZn). Between the two materials, NiZn has higher permeability ($\mu_r = 8.11 + j2.27$ at 100 MHz), which provides higher inductance values at reduced sizes, which decreases ripple current and DC resistance simultaneously and increases the efficiency. On the other hand, CIP has lower magnetic loss tangent ($\mu_r = 5.64 + j0.57$ at 100 MHz), which results in increased efficiency by reducing AC loss in Equation 5.15. To accurately account for magnetic material effect on IVR efficiency, measured frequency dependent complex permittivity and permeability of both CIP and NiZn from [76] are imported into the EM solver for accuracy.

Table 5.4: Control Parameters of Solenoidal Inductor

Parameter		Unit	Min	Max
Gap between windings	g	mil	2	20
Number of windings	N		3	13
Size of via	s_v	μm	50	103
Copper Trace Width	w_c	mil	2	20
Copper Thickness Bottom	$t_{c,b}$	μm	35	170
Copper Thickness Top	$t_{c,t}$	μm	35	170
Dielectric Thickness	t_d	μm	50	650
Dielectric Width	w_d	μm	50	350
Magnetic Core Thickness Ratio	t_m		0.1	1
Magnetic Core Width offset	Δw_m	mil	0	100

5.5.3 Optimization Setup

Among the components of the IVR, the focus in this work is on the optimization of inductor to maximize IVR efficiency. Ten control parameters are defined for the inductor as shown in Table 5.4 along with the range used for each parameter based on process capabilities. The objective function used for optimization is

$$f(x) = \sum_{i=1}^{2} w_i y_i \tag{5.19}$$

where y_1 and y_2 are peak overall IVR efficiency calculated using the described model and area of inductor respectively; $w_1 = 5$ and $w_2 = -2$ are corresponding weights of multi-objective optimization. Since IVR efficiency is the main goal of the optimization process, it has higher weight compared to the inductor area. By choosing such weights, TSBO adopts the control parameters providing 2% of power efficiency over 5mm^2 of the inductor area.

To investigate opportunities for reducing fabrication costs, two types of inductors are analyzed: one that uses only 1 oz copper (**Type I**) and the more costlier option that uses copper thickness up to $170\mu m$ (**Type II**). In the case of Type II inductor, the optimization is done with 10 parameters and for Type I, only 8 parameters are used since $t_{c,b}$ and $t_{c,t}$ are fixed to $35\mu m$.

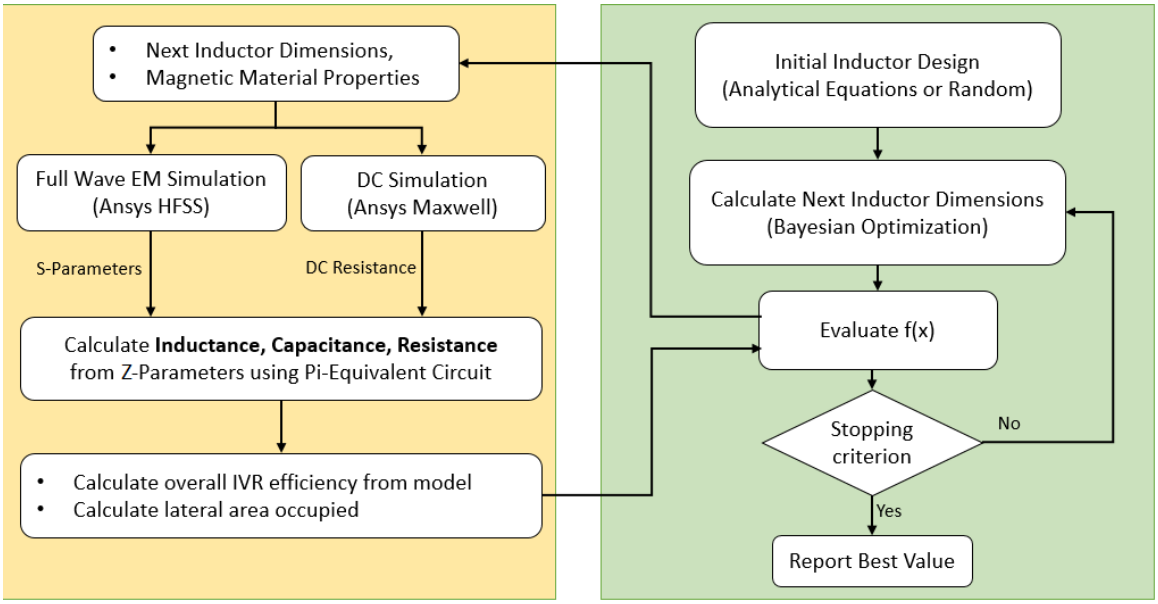


Figure 5.10: Automated optimization setup used in IVR application

The automated optimization setup used is given in Figure 5.10. Inductor dimensions are determined using TSBO and fed into the full-wave solver to extract the 2-port Z-matrix, which is then used by the inductor and buck converter models for calculating IVR efficiency. Calculated efficiency is combined with inductor area in Equation 5.19 and fed back into TSBO to proceed to the next iteration.

5.5.4 Results

To make a direct comparison with hand-tuned design in [74], power efficiencies are calculated assuming R_{PDN} = 10 mΩ and ESR_C = 10 mΩ in Equation 5.17 and Equation 5.16. Among four optimized IVRs, the one using Type II inductor with CIP showed the best performance, providing efficiency of 85.1%, 93.1% and 94.1% for 5V:1V, 3V:1V and 1.7V:1.05V conversions respectively, with an inductor area of 5.1 mm^2. Compared to hand-tuned design in [74], the efficiency is increased by 5.7%, 4.5% and 3% accompanied with 55.3% reduction in inductor area.

On the other hand, the cheaper option, IVRs with Type I inductors showed comparable efficiency with hand-tuned design but with 48.2% reduced inductor area. In this case, IVR

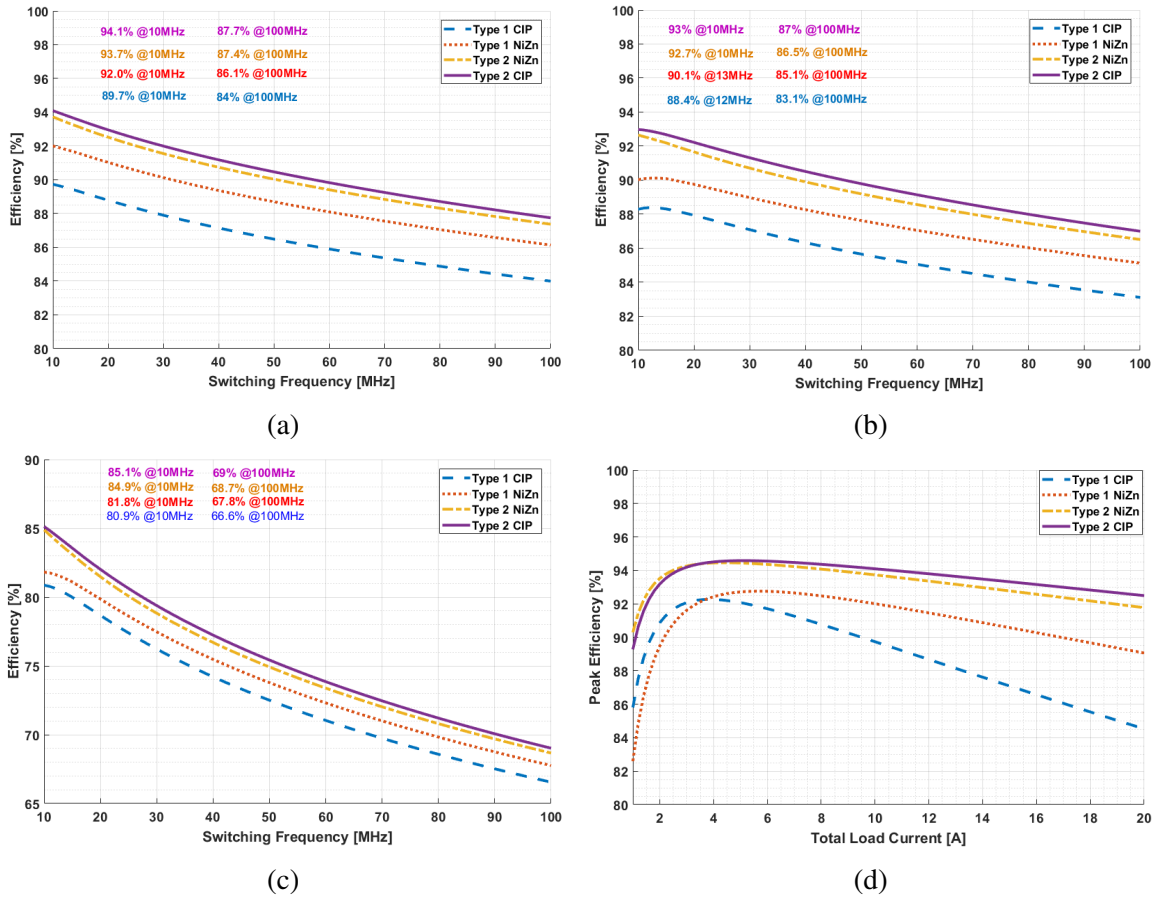


Figure 5.11: Efficiency comparison of four optimized IVRs as a function of switching frequency and total load current. (a) 1.7V:1V Conversion (I_{LOAD}=10A). (b) 5V:1V Conversion (I_{LOAD}=10A). (c) 5V:1V Conversion (I_{LOAD}=10A). (d) Eff. vs Load current (f_{SW}=10 MHz)

using inductor with NiZn core outperformed the inductor with CIP. This shows that when copper thickness is limited, using materials with higher permeability as compared to lower magnetic loss tangent provides more opportunities to increase IVR efficiency by decreasing number of windings and hence, DC loss in Eq. Equation 5.14.

Comparison for power efficiencies of each optimized IVR as a function of switching frequency at I_{LOAD} = 10 A (2.5A per phase) and as a function of total load current at f_{SW} = 10 MHz are given in Figure 5.11. In addition, Table 5.5 shows the inductor characteristics of each optimized IVR as well as peak conversion efficiencies and compares it to the hand tuned design from [74].

Figure 5.12 compares the performance of TSBO to non-linear solver, GP-UCB and

Table 5.5: Comparison of Optimized IVRs and Inductors to Hand Tuned Design

	Hand Tuned [74]	**Type I NiZn**	**Type I CIP**	**Type II NiZn**	**Type II CIP**
L [nH]	24.8	13.32	15.4	29.4	23.47
R_{AC} [Ω]	2.27	0.87	0.51	2.67	0.98
R_{DC} [mΩ]	14.7	17.0	30.1	10.5	8.7
Area [mm^2]	11.6	6.0	6.1	5.2	5.1
Peak eff. 5V:1V	79.4%	81.8%	80.9%	84.9%	85.1%
Peak eff. 3V:1V	88.5%	88.4%	90.1%	92.7%	93.0%
Peak eff. 1.7V:1.05V	91.1%	92.0%	89.7%	93.7%	94.1%

IMGPO for the objective of maximizing Equation 5.19, along with the corresponding value of hand-tuned design. Optimization using TSBO resulted in 85.1% peak efficiency for 5V:1V conversion with the inductor area of 5.16 mm^2, compared to 78.6%, 84.4% and 84.9% with 25.2 mm^2, 6.64 mm^2 and 5.18 mm^2 for non-linear solver, IMGPO and GP-UCB, respectively. Though all algorithms started from the same initial point, TSBO reached the pre-determined error tolerance in Figure 5.12 using 27 simulations (51.1 minutes), compared to 60 and 59 simulations (115.6 and 117.3 minutes) for IMGPO and GP-UCB, corresponding to a reduction of 56.7% and 57.4% in CPU time. Besides, optimization using non-linear solver resulted in 78.6% peak efficiency with the inductor area of 25.2mm^2 and could not reach the error tolerance within 100 simulations (185 minutes).

Figure 5.13 shows the breakdown of objective function in Equation 5.19 to its two components of peak efficiency and inductor area. This breakdown shows that the TSBO converges around 30 simulations resulting in 84.5% efficiency with 5.78mm^2 area, whereas IMGPO and GP-UCB converges around 60 simulations. This shows that the number of simulations required for each algorithm to cross the tolerance level in Figure 5.12 coincides with their convergence point, hence, validates the CPU time comparison made in Table 5.6.

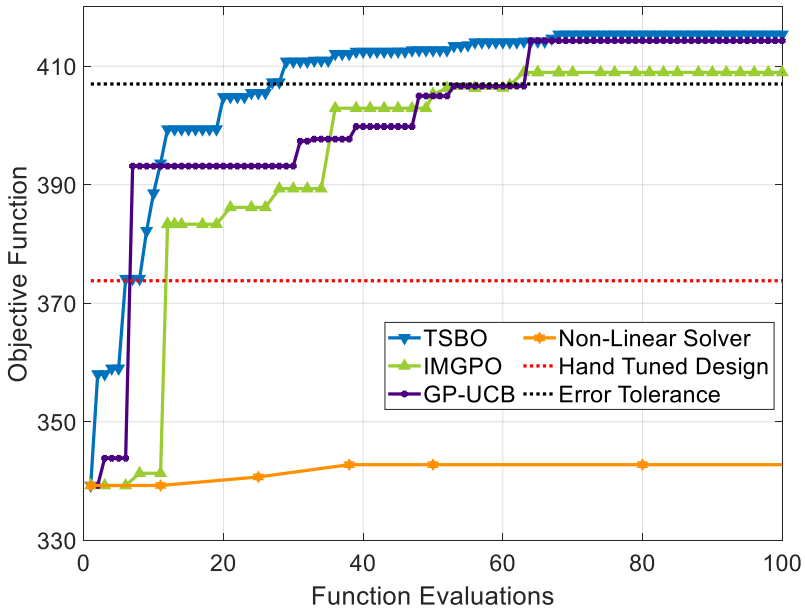
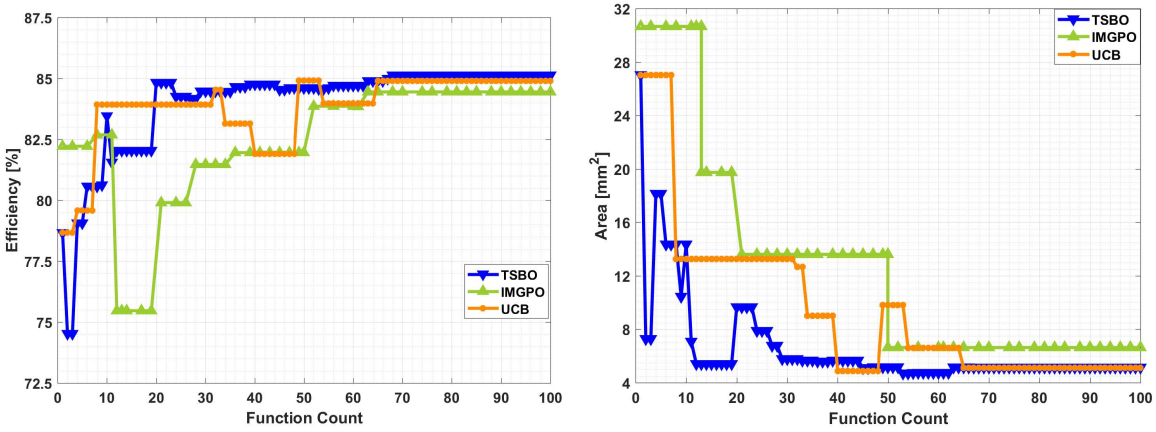


Figure 5.12: Performance comparison of TSBO to non-linear solver, GP-UCB and IMGPO on maximizing objective function at Equation 5.19



(a) (b)

Figure 5.13: Convergence comparison for the two components of objective function in Equation 5.19

Table 5.6: Optimization Results for IVR using Type II Inductor with CIP

	Non-Linear Solver	**GP-UCB**	**IMGPO**	**TSBO**
Peak eff. 5V:1V (f_{SW}=10MHz, I_{Load}=10A)	78.6 %	84.9 %	84.4 %	85.1 %
Area	25.2 mm^2 (+79.6%)	5.18 mm^2 (+0.4%)	6.64 mm^2 (+22.3%)	5.16 mm^2
CPU Time	> 185 min (> +72.9%)	117.3 min (+57.4%)	115.6 min (+56.7%)	50.1 min

5.6 Conclusion

In this chapter, we have presented a new BO based global optimization algorithm, *Two Stage Bayesian Optimization*, that considers EDA related challenges in optimization. Unlike conventional BO algorithms, the proposed algorithm is not prone to initial point selection and does not rely on other algorithms for auxiliary optimization. This was achieved by using a new hierarchical partitioning scheme that makes the algorithm EDA oriented in terms of substantially reducing number of simulations required to reach the global optima in large sample spaces. Furthermore, we have presented a strategy to learn which acquisition function best suits to the given optimization problem, making TSBO applicable to various design problems rather than being specific to a particular type. Empirical analysis on a set of popular challenge functions with several local extrema and dimensions showed TSBO to have a faster convergence trend over other widely used methods.

Furthermore, we have showed how machine learning, in particular, TSBO enables enhanced designs for two emerging applications, namely 3D ICs and IVRs. In the IVR application, optimized IVR with Type II inductor with CIP using TSBO resulted in peak IVR efficiency for 5V:1V (at I_{LOAD} = 10A and f_{SW} = 10MHz) of 85.1% with the embedded solenoidal inductor occupying an area of 5.1mm^2 corresponding to 5.7% increase in efficiency and 56.1% reduction of area compared to hand tuned design. Moreover, TSBO showed 72.4%, 57.4%, 56.7% reduction in CPU time required to complete optimization

compared to non-linear solver, GP-UCB and IMGPO respectively. This trend of faster convergence of TSBO was also observed in application for clock skew minimization of 3D ICs, where TSBO was shown to be 3.76 and 3.96 times faster than IMGPO and non-linear solver. The results presented in this chapter show the proposed algorithms applicability to a variety of systems that can be represented as a black box, and hence the proposed method is very general.

CHAPTER 6

HIGH-DIMENSIONAL BAYESIAN OPTIMIZATION FOR HIGH-FREQUENCY ELECTRONIC DESIGN

Efficient global optimization of microwave systems is a very challenging task that emerges in importance for rapid design closure and discovery of novel structures. As the operating frequency increases, however, such systems tend to become more sensitive to variations and interactions between their control parameters, such as geometrical dimensions and material properties. This raises new challenges related to design optimization, especially when the number of such parameters increases due to interactions and non-linearity. Approximations made using analytical models often tend to provide erroneous results and unable to capture these interactions. Accurate models used in optimization problems therefore need to rely on accurate electromagnetic (EM) simulations. Since such simulations are computationally expensive, the problem is further acerbated. From the optimization perspective, the solution therefore corresponds to minimizing or maximizing a non-convex, high-dimensional response surface where the function queries are generated from an EM simulator. Since the analytical form of the response surface is not available, the optimization needs to be conducted in the black-box setting.

A promising method with strong theoretical foundations to address these challenges is Bayesian Optimization (BO) techniques based on Gaussian Processes (GP) as in previous chapter. However, scaling BO method to higher-dimensional problems ($D > 10$) in an efficient manner is a challenging and open problem in the literature.

In this chapter, we propose a new method to scale BO to high-dimensional problems, namely Bayesian Optimization with Deep Partitioning Tree (DPT-BO). In this method, the objective is to reduce the computational complexity through a reduction in the number of EM simulations required for the optimization. Motivated by recent advances in high-

dimensional BO (HDBO) [77, 78, 79], we adopt the use of Additive Gaussian Processes (ADD-GP) and approximate the objective function as a summation over lower dimensional group of functions, each depending on a subset of input parameters [80]. Previous methods utilizing ADD-GP assume the function decomposes into non-overlapping groups, meaning that the parameters in a particular group can be optimized independently without considering the parameters in other groups. This is mainly to handle computational complexity associated with the auxiliary optimization of the non-convex acquisition function used in the BO framework. However, non-overlapping groups make use of an assumption that is invalid in high frequency design in that high dimensional systems tend to have interacting variables. For instance, geometrical parameters of an antenna and its matching network can not be optimized independently from each other. Hence, we propose using a fully additive decomposition where all interactions at a particular order are preserved, allowing to capture much richer classes of functions. To the best of our knowledge, the fully additive structure that allows for capturing interactions between every parameter has not been reported in the context of BO in the open literature.

To address the challenge of auxiliary optimization of the acquisition function, we extend the hierarchical partitioning tree approach presented in Chapter 3 of this thesis and propose a novel *deep hierarchical partitioning tree*. The proposed strategy eliminates the auxiliary optimization step from the BO framework and allows for a rapid coverage of the high-dimensional sample space. We show the performance DPT-BO on several optimization test functions consisting of many local optima and different response surfaces, as well as three applications arising in high frequency electronic design, namely 1) maximizing signal integrity in high-speed channels, 2) minimizing losses over D-band for substrate integrated waveguides (SIW) with an air cavity and 3) maximizing efficiency of a wireless power transfer (WPT) system. The MATLAB implementation of DPT-BO used in these applications are publicly available at `https://github.com/hakkimerttorun/DPTBO`.

6.1 Bayesian Optimization with Deep Partitioning Tree

There are two main challenges in efficiently scaling BO to higher dimensional problems: 1) statistical challenge of approximating the objective function, $f(x)$, and 2) auxiliary optimization of the non-convex acquisition function. The former is related to the need for exponentially more observations to establish a certain accuracy of the non-parametric GP model, i.e. curse of dimensionality. The latter is a computational challenge since this auxiliary optimization itself is exponentially difficult in higher dimensions. Addressing these two challenges efficiently requires making assumptions on the structure of $f(x)$ to efficiently handle the optimization complexity.

In this section, we first present the structural form of $f(x)$ we use in DPT-BO and compare with the existing forms in literature. Then, we present the *Deep Partitioning Tree* (DPT) as a sampling strategy to select the next system simulation parameters and address the auxiliary optimization challenge.

6.1.1 Model Structure

A substantial portion of the prior work in HDBO assumes the objective function lies in a lower dimensional subspace of the original sample space. Here, attempts are mainly based on approximating the original function with a lower dimensional embedding [81, 82]. However, this is a very strict assumption for high-frequency design optimization since the high dimensionality can arise due to a combination of device and circuit level components' parameters such as transistor size, lumped element parameters and geometrical parameters of EM structures, all of which significantly affect the system performance.

Recently, this assumption was relaxed by assuming that $f(x)$ admits an additive form, thus, allowing it to depend on all input variables. Functions of this form can be written as [77]

$$f(x) = f^{(1)}(x^{(1)}) + f^{(2)}(x^{(2)}) + ... + f^{(M)}(x^{(M)}) \tag{6.1}$$

where $x^{(j)}$ represent parameters that are decomposed into M disjoint groups such that $x = \bigcup_{j=1}^{M} x^{(j)}$ and $x^{(j)} \cap x^{(i)} = \emptyset$. This structure can be realized at the kernel level of the GP by parameterizing the kernel function as:

$$k(x, x') = k^{(1)}(x^{(1)}, x'^{(1)}) + k^{(2)}(x^{(2)}, x'^{(2)}) + \cdots + k^{(M)}(x^{(M)}, x'^{(M)}) \tag{6.2}$$

where $k(x, x')$ is the D dimensional kernel used to construct the covariance matrix in Equation 2.6. Note that the structure in Equation 6.1 simplifies the auxiliary optimization of the acquisition function. Since $u(x)$ admits the same structure, each group of $u^{(j)}(x^{(j)})$ can be optimized separately. Several works are built on this formulation to efficiently learn the decomposition [78] or relax the disjoint assumption by allowing a limited degree of interaction between groups [79]. However, the underlying structural assumption in Equation 6.1 is still conservative since grouping the input variables in such a way means that $f(x)$ can be optimized by finding the optimal parameters in each group independently from the parameters in other groups, which is hardly the case in high-frequency designs.

In this chapter, we further relax the structural assumption and propose using a fully additive decomposition to preserve the interactions between variables. Here, the additive structure we consider corresponds to *high-dimensional model representation* (HDMR) [83] and can be written as:

$$f(x) = \sum_{i=1}^{D} f_i(x_i) + \sum_{1<i<j<D} f_{ij}(x_i, x_j) + \sum_{\substack{1<i_1 \\ ...<i_n<D}} f(x_{i_1}, ..., x_{i_n}) \tag{6.3}$$

where n denotes the maximum allowed order of interactions among D possible orders. Since the GP we use has a constant mean function, the structure in Equation 6.3 can be

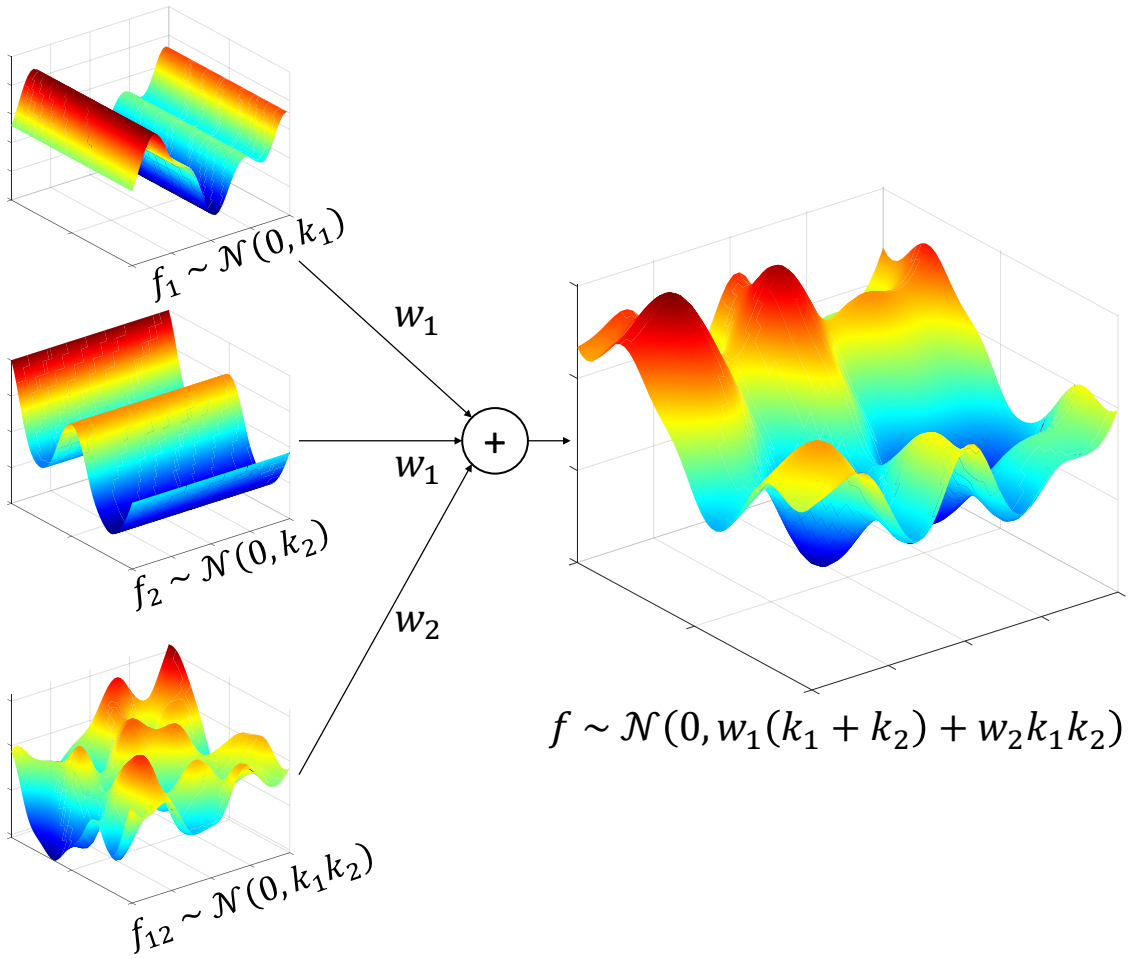


Figure 6.1: 2D illustration of the additive decomposition used by DPT-BO.

realized via the fully additive kernel of ADD-GP, written as [80]

$$\begin{aligned} k(x, x') =& \sigma_{f_1}^2 \sum_{i=1}^{D} k_i(x_i, x_i') + \sigma_{f_2}^2 \sum_{1<i<j<D} k_i(x_i, x_i') k_j(x_j, x_j') \\ & \cdots + \sigma_{f_n}^2 \sum_{\substack{1<i_1 \\ \ldots<i_n<D}} \left[\prod_{d=1}^{n} k_{i_d}(x_{i_d}, x_{i_d}') \right] \end{aligned} \tag{6.4}$$

where $k_i(x_i, x_i')$ is the sub-kernel of each parameter, chosen as Matern 5/2 as in Equation 2.8 with unit variance; $\sigma_{f_n}^2$ is the signal variance hyperparameter associated to each order of interaction, enabling to learn the contribution of each level of interaction to the overall predictive GP. It should also be noted that as each input variable has a separate ker-

nel, a separate lengthscale is assigned to every parameter, effectively implementing ARD. The prior associated with the kernel function in Equation 6.4 is illustrated in Figure 6.1 for the 2D case.

The number of terms in the right-hand side of Equation 6.3 increases super-exponentially with D. Hence, we only preserve up to R^{th} order of interactions in Equation 6.4) along with the D^{th} order of interaction corresponding to a standard GP model, i.e. $n = 1, 2, ..., R, D$. This makes the proposed DPT-BO algorithm a superset of the BO methods using a standard GP. Here, during the training of the GP model, the likelihood in Equation 2.9 can favor increasing σ_D^2 and decreasing $\sigma_{1,...,R}^2$ for a particular problem and vice versa for others. As the proposed DPT-BO algorithm updates the hyperparameters at every iteration, this corresponds to dynamically weakening or strengthening the structural assumption in Equation 6.3 as the optimization progresses and allows for considering much richer classes of functions compared to prior work in literature.

6.1.2 Deep Hierarchical Partitioning Tree

A promising direction to address the second challenge of HDBO, the auxiliary optimization of the acquisition function, is to use the hierarchical partitioning tree approach of TSBO as presented in Chapter 3 of this thesis. However, the approach is not directly applicable to high-dimensional problems. The sub-regions generated by the partitioning tree increases in volume when we consider more parameters. Hence, to obtain a small enough region, we need to expand more child nodes, corresponding to more function queries. Further, as each expansion generates 2^D candidate points, the algorithm can require a significant memory for high-dimensional problems ($D > 10$).

In this chapter, we therefore build on the candidate region generation strategy of TSBO with a novel deep partitioning tree. Unlike conventional partitioning structures where the child nodes are only expanded in the vertical direction, the proposed strategy also performs expansion in the horizontal direction, thus, creating a deep structure.

Here, at every iteration, we group the input parameters according to their sensitivity of creating a variation on $f(x)$. As the kernel we use in Equation 6.4 has the ARD property, the input variables having a lower lengthscale value have higher impact on $f(x)$ based on the correlation matrix in Equation 2.6. This provides a cost-free impact analysis which can be used to perform the grouping as

$$\left\{ x^D = \bigcup_{j=1}^{M} x_t^{(j)} \;\middle|\; |x_t^{(j)}| = d \ll D \right\} \tag{6.5}$$

and

$$\left\{ \lambda^D = \bigcup_{j=1}^{M} \lambda_t^{(j)} \;\middle|\; \max \lambda_t^{(i)} < \min \lambda_t^{(j)},\ \ \forall i < j < M \right\} \tag{6.6}$$

where x^D is the D dimensional input vector; $x_t^{(j)}$ is the j^{th} group of input variables at t^{th} iteration; M is the total number of groups; d is the maximum number of parameters in a particular group and $\lambda_t^{(j)}$ represents the lengthscale vector of the input parameters in the j^{th} group. Note that as the training of the GP is done at every iteration and the hyperparameter vector is updated, input variables in each group changes dynamically according to Equation 6.6.

Once the input parameters are grouped, the sample space X^D is divided into 2^d regions of hypercubes, $H_{t,k}$, along each parameter in a particular group, $x_t^{(j)}$. We refer to this division as the *vertical expansion* of the partitioning tree, which is repeated for each group, resulting in a total of $M2^d$ new regions at every iteration. Then, candidate points are generated as:

$$c_{t,k} = \frac{H_{t,k,min} + H_{t,k,max}}{2}, \quad k = 1, 2, ..., tM2^d \tag{6.7}$$

where $c_{t,k}$ is the k^{th} candidate point of a total of $tM2^d$ points after iteration t; $H_{t,k}$ is the region that $c_{t,k}$ belongs to and $H_{t,k,min}$ and $H_{t,k,max}$ are the corresponding lower and upper boundaries of each D dimensional region.

After the vertical expansion, a possible sampling strategy can be to select the next

sampling point as the candidate point maximizing the acquisition function. However, the candidate points generated after the vertical expansion belong to a relatively larger region. Such an approach would provide fast exploration of the sample space since it identifies the most promising region among many other large regions that comprise the high-dimensional sample space. Yet, it would lack exploitation property since a subsequent search within this large promising region is not performed. This would limit the convergence rate in high-dimensional problems since more simulations would be required to explore a significant portion of the sample space such that the identified promising region becomes small enough.

To address the exploitation problem without giving up the exploration capability, we propose a horizontal expansion strategy following the vertical expansion to search within these relatively large regions. First, we select the most promising region that is generated after the vertical expansion as:

$$c_{t,j^*} = \arg\max_{c \in C} u(c), \quad c_{t,j^*} \in \widetilde{H}_t \tag{6.8}$$

where C is the set of previously generated candidate points; $u(c)$ is the value of the acquisition function and $\widetilde{H}_t$ is the relatively large region that c_{t,j^*} belongs to. We then iteratively shrink $\widetilde{H}_t$ by dividing it into 2^d regions along the parameters in a particular group, $x_t^{(i)}$, and generate the candidate points within these regions as:

$$H_t^{(i)} = \bigcup_{j=1}^{2^d} h_{t,j}^{(i)}, \quad c_{t,j}^{(i)} = \frac{h_{t,j,min}^{(i)} + h_{t,j,max}^{(i)}}{2} \tag{6.9}$$

where $h_{t,j}^{(i)}$ and $c_{t,j}^{(i)}$ denote the resulting regions and corresponding candidate points when $H_t^{(i)}$ is divided into 2^d new regions. This is followed by selecting the most promising region, $h_{t,j^*}^{(i)}$, from these newly generated ones as in Equation 6.8 and further dividing it into 2^d new regions along the parameters in the next group, $x_t^{(i+1)}$, as in Equation 6.9.

Algorithm 1 Horizontal Expansion in DPT-BO.

1: (Initialize) $H_t^{(1)} \leftarrow \widetilde{H}_t$, $i \leftarrow 1$.
2: $\left(\text{Update } H_t^{(i)}\right)$ $H_t^{(i)} \leftarrow \bigcup_{j=1}^{2^d} h_{t,j}^{(i)}$ along $\forall x \in x_t^{(i)}$
3: $\left(\text{Update } c_{t,j}^{(i)}\right)$ $c_{t,j}^{(i)} \leftarrow \left(h_{t,j,min}^{(i)} + h_{t,j,max}^{(i)}\right)/2$,
4: $\left(\text{Select } c_{t,j^*}^{(i)}\right)$ $c_{t,j^*}^{(i)} \leftarrow \arg\max_{c \in C} u(c)$
5: $\left(\text{Set } H_t^{(i+1)}\right)$ $H_t^{(i+1)} \leftarrow h_{t,j^*}^{(i)}$
6: **if** i = M **then**
7: $x_{t+1} \leftarrow c_{t,j^*}^{(M)}$
8: **else**
9: $i \leftarrow i + 1$ and go to Step 2.
10: **end if**

The horizontal search within $\widetilde{H}_t$ is performed M times starting from the first group, i.e. $i = 1$. This corresponds to a sensitivity aware search that prioritizes the input parameters having more impact on $f(x)$ since $x_t^{(i)}$ have more impact on $f(x)$ than $x_t^{(i+1)}$ according to Equation 6.6 and $x_t^{(i)}$ determine the region to be further divided along $x_t^{(i+1)}$. Note that, at t^{th} iteration, the total number of candidate points generated by DPT-BO corresponds to $2tM2^d$ after the vertical expansion followed by the horizontal one.

Finally, the next sampling point is selected as the candidate point that provides the largest acquisition function value after M^{th} division as:

$$x_{t+1} = \underset{c \in h_{t,j}^{(M)}}{\arg\max}\, u(c) \tag{6.10}$$

This horizontal expansion procedure can be implemented using a recursive formulation as given in Algorithm 1. An example partitioning tree constructed by DPT-BO is given in Figure 6.2.

After performing the simulation at x_{t+1}, the region that x_{t+1} belongs to, $h_{t,j^*}^{(M)}$, is verti-

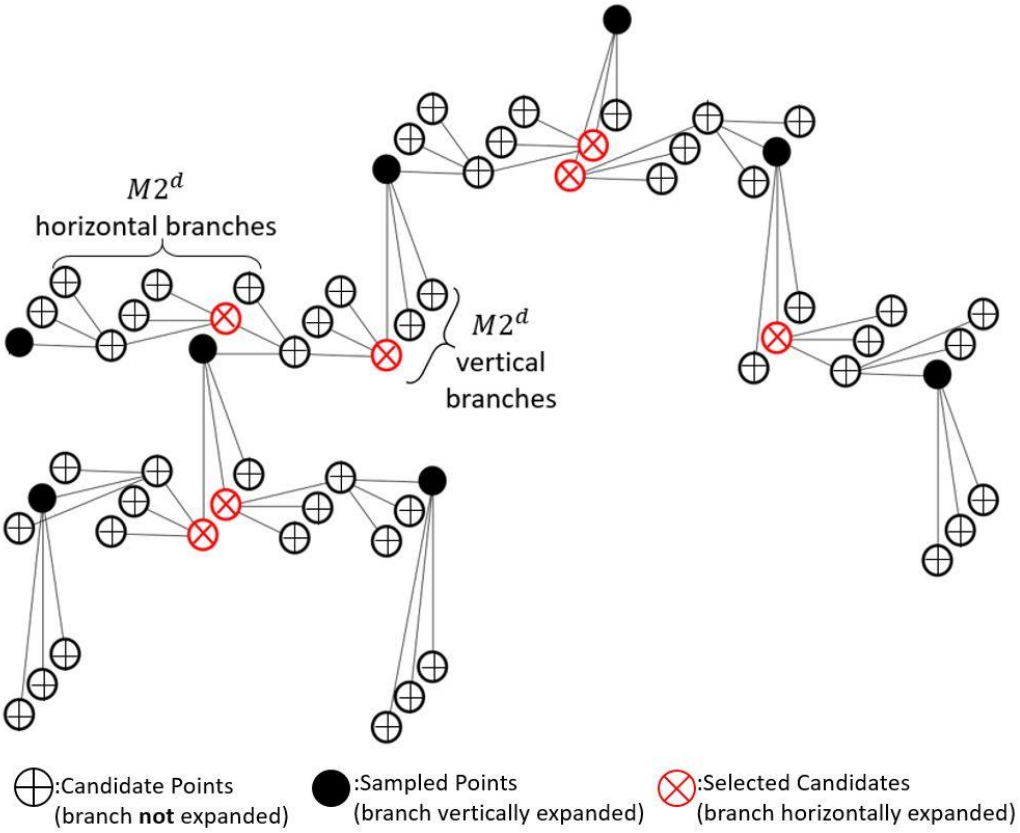


Figure 6.2: An example Deep Partitioning Tree (DPT)

cally expanded to proceed into next iteration as:

$$H_n = \bigcup_{i=1}^{M2^d} h_i, \quad h_i \subset h_{(t,j^*)}^{(M)} \tag{6.11}$$

$$H_{t+1} = H_n \cup H_t \tag{6.12}$$

where H_n is the union of new regions, h_i, acquired by dividing $h_{t,j^*}^{(M)}$ into $M2^d$ hypercubes.

6.1.3 Selecting Acquisition Function

Determining which acquisition function to use is a vital part of any BO based algorithm. In recent years, information-theoretic entropy based acquisition functions, such as predictive entropy search (PES) [84], have gained particular interest in BO community. Such methodologies aim to directly estimate *argmax* of the function and is shown to be particu-

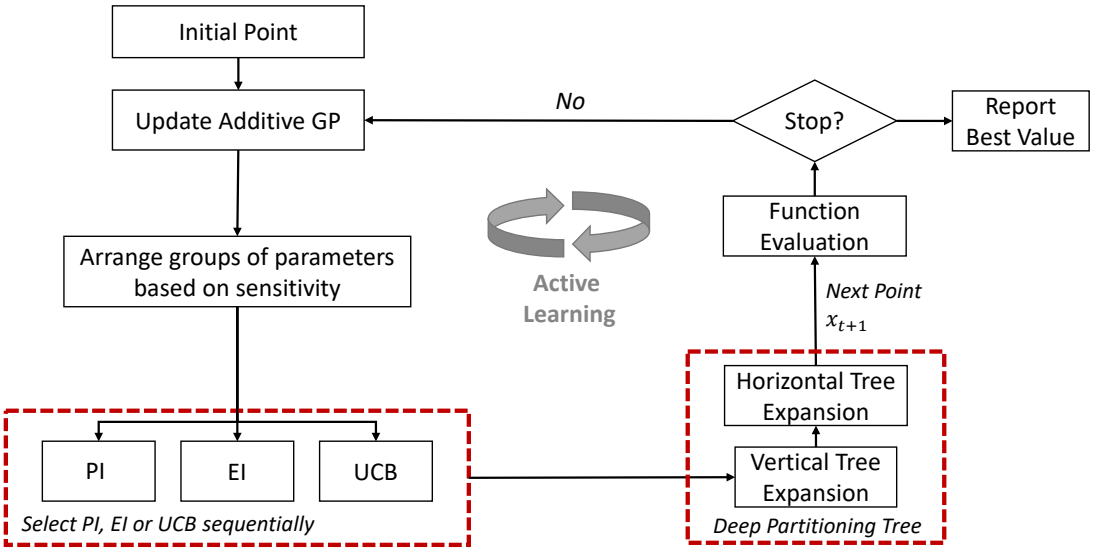


Figure 6.3: Overall flowchart of the proposed DPT-BO method.

larly useful as compared to improvement based probability of improvement (PI), expected improvement (EI) and upper confidence bound (UCB) for problems with moderate dimensionality. However, for high-dimensional problems, information-theoretic approaches become computationally intractable since such methodologies involve estimating intractable distributions. This requires either making poor approximations, such as mean-field approximation, that can result in poor convergence or resorting to sampling based techniques that introduce additional computational complexity.

In DPTBO, we therefore use improved based acquisition functions. As explained in Chapter 3, there is no guarantee that a single improvement based strategy will outperform others at every iteration and for every problem. However, a reward-based strategy to choose which acquisition function to use as the learning acquisition functions strategy presented of TSBO in Chapter 5 can be misleading in high-dimensional problems since the reward function can be biased towards a particular $u(x)$ in the early iterations of the optimization. Since this bias can propagate into subsequent iterations, overall convergence rate can degrade. Hence, in DPT-BO, we use PI, EI and UCB in a sequential manner to benefit from each strategy without running into the biasing problem. The overall flowchart summarizing the DPT-BO method is given in Figure 6.3.

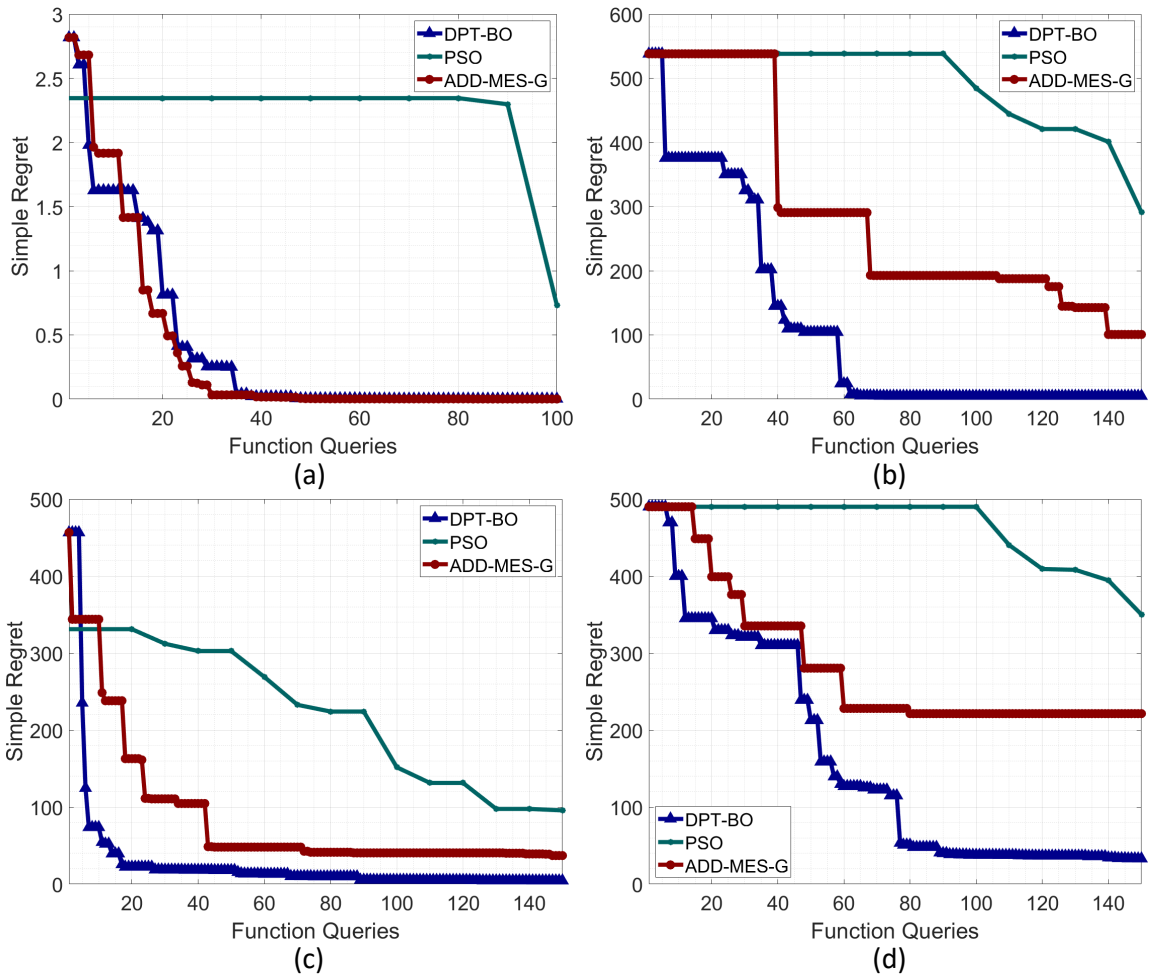


Figure 6.4: Performance comparison of the proposed algorithm, DPT-BO, on optimization test functions. (a) 6D Hartmann. (b) 10D Schwefel. (c) 15D Levy. (d) 25D Qing. Functions are available in [68].

6.1.4 Experiments on Test Functions

In order to test the performance of DPT-BO, we have considered four test functions with different dimensionalities and response surface types that contain many local optima, which are common benchmarking problems in black-box optimization [68]. Performance evaluation criteria in these experiments are 1) *simple regret* (r_t), the distance between actual global optima and the one found by the optimization algorithm and 2) area under curve

Table 6.1: Performance of DPT-BO on Test Functions

	PSO		ADD-MES-G		DPT-BO	
	r_t	AUC (Normalized)	r_t	AUC (Normalized)	r_t	AUC (Normalized)
6D Hartmann	0.730	5.11	**0.001**	**0.62**	0.002	1.00
10D Schwefel	291.4	4.32	100.7	2.61	**4.768**	**1.00**
15D Levy	95.74	7.57	37.07	3.10	**4.514**	**1.00**
25D Qing	350.0	2.88	221.3	1.65	**33.09**	**1.00**

Table 6.2: Run Times for Selecting the Next Sampling Point

	PSO	ADD-MES-G	DPT-BO
6D Hartmann	0.028s	0.324s	0.379s
10D Schwefel	0.032s	0.832s	1.205s
15D Levy	0.034s	1.701s	1.348s
25D Qing	0.034s	4.561s	2.501s

(AUC), a measure of how fast an algorithm converges. AUC and r_t can be written as:

$$r_t = |f^* - \widetilde{f}_t^*|, \quad AUC = \frac{1}{T}\sum_{i=1}^{T} r_t \tag{6.13}$$

where f^* is the known global optimum of $f(x)$ and $\widetilde{f}_t^*$ is the best value found by the optimization algorithm at t^{th} iteration.

We compare DPT-BO with the widely used particle swarm optimization (PSO) and a recent high-dimensional BO method, maximum entropy search with additive GP (ADD-MES-G) [85], that uses the kernel in Equation 6.2 along with a structural kernel learning strategy [78].

Results of these experiments are provided in Figure 6.4 and in Table 6.1. Except for the 6D Hartmann function, the proposed DPT-BO algorithm results in a significantly better regret value than both ADD-MES-G and PSO. In addition, normalized AUC values show that, on average, DPT-BO converges 1.99X and 4.97X faster than ADD-MES-G and PSO, respectively. Table 6.2 further provides CPU times required to select the next sampling point for each technique. As PSO does not use a surrogate model, its run time is considerably faster compared to both ADD-MES-G and DPT-BO. Although there is no significant difference between run times of ADD-MES-G and DPT-BO, the sampling strategy of ADD-MES-G for structural kernel learning and estimating the acquisition function makes it slower as compared to DPT-BO. Note that for these experiments, the hyperparameters of DPT-BO are chosen as: $\eta = 0.05$ in Equation 2.20; $\zeta = 0.01$ and $\zeta = 0.1$ in Equation 2.18 and Equation 2.19, respectively; $n = 1, 2, D$ in Equation 6.4 and $d = 3, 5, 5, 5$ and $M = 2, 2, 3, 5$ in Equation 6.5 for 6D Hartmann, 10D Schwefel, 15D Levy and 25D Qing functions, respectively.

In order to quantify the effect of number of groups and maximum number of parameters in a group (d and M in Equation 6.5) on the performance of DPT-BO, an additional experiment is performed on the 15D Levy function. Here, d is varied from 1 to 8 with the corresponding M values varying between 15 to 2. Results of this experiment are given in Figure 6.5. The convergence is observed to be the worst when $d = 1$ and $d = 8$ due to over exploration for the former case and over exploitation for the latter as explained in previous subsections. The convergence for the $d = 5$ case is observed to be better compared to the case with $d = 3$. The reason is when d increases, the sensitivity of parameters becomes less important in the horizontal expansion strategy. This provides an advantage since the prioritization is less affected by the bias introduced in early iterations of the optimization process, when the number of observations are not sufficient enough to accurately group the parameters. It should be noted that since the acquisition functions used already address the exploration & exploitation trade-off, all choices of d and M have a certain level of balance.

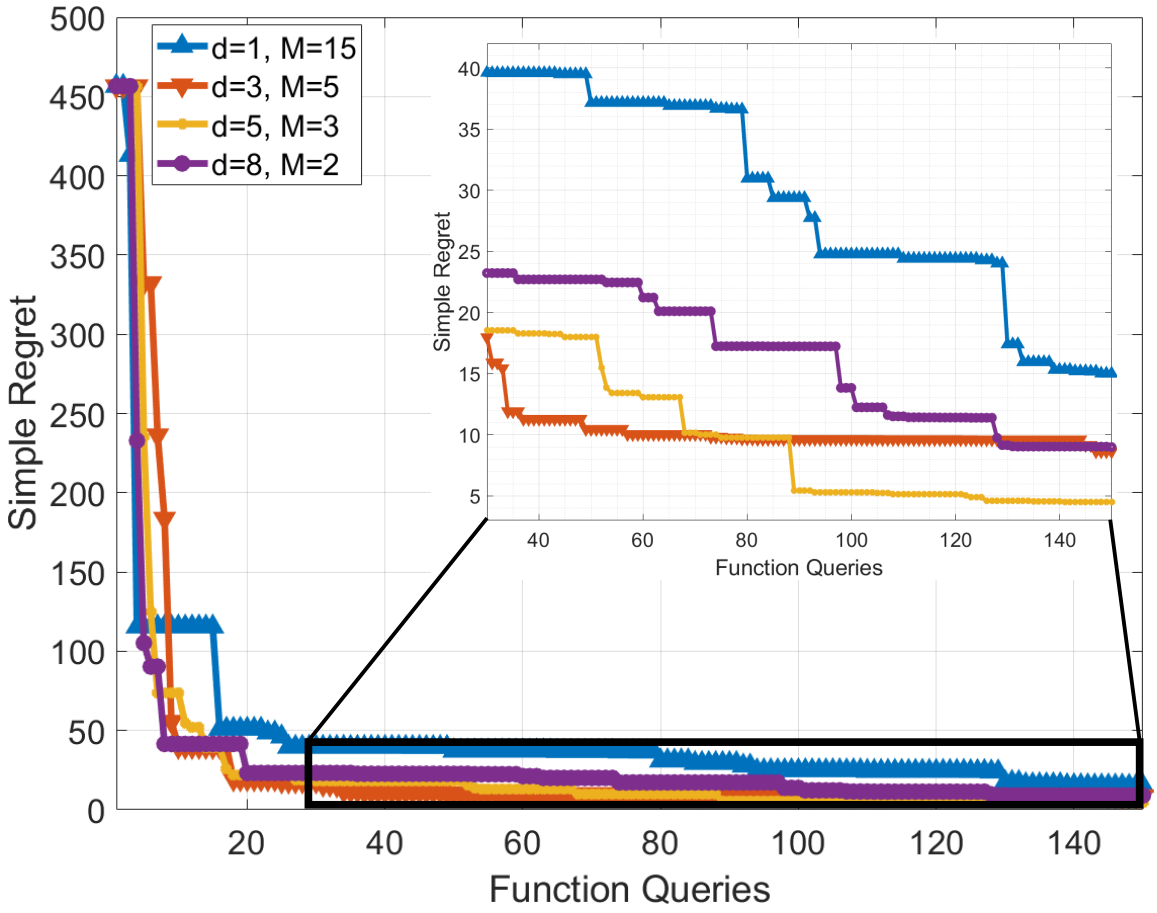


Figure 6.5: Effect of number of groups (d and M in Equation 6.5) on performance of DPT-BO on minimizing 15D Levy function.

The parameters d and M thus can not provide pure exploitation or pure exploration. Following this analysis, we choose $d = 3, 5, 6$ and $M = 3, 3, 5$ for the design applications in Sections 6.1, 6.2 and 6.3 of this chapter, respectively.

6.2 Application 1: Interconnects in High-Speed Channels

The first application chosen to evaluate the DPTBO algorithm is optimization of interconnects in high-speed chip-to-chip interconnect channels. Optimization of such high speed channels is a non-trivial task given that the system has a non-linear response to its control parameters such as the geometrical parameters of the interconnects and the material properties. Furthermore, characterizing electrical properties of such interconnects requires

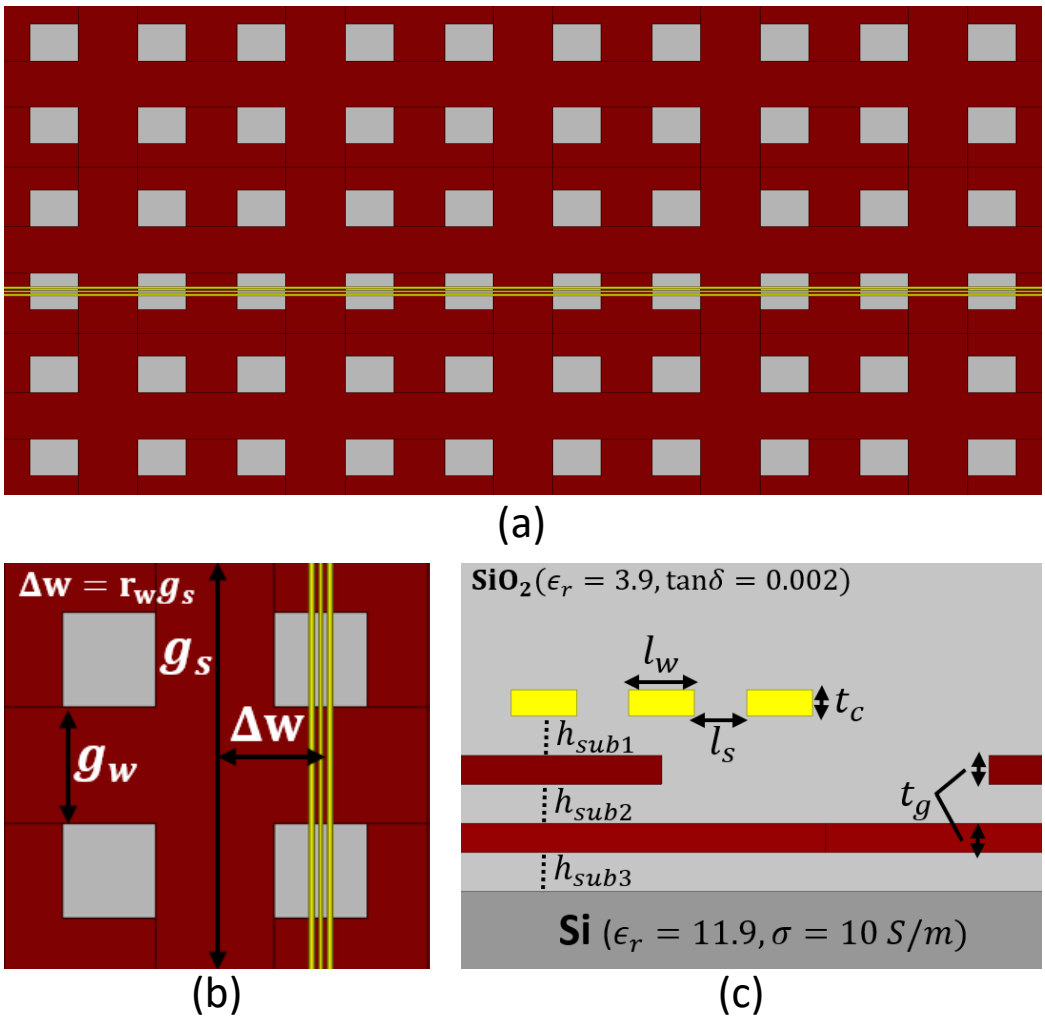


Figure 6.6: Structure of the high-speed channel considered. (a) Top view of overall channel. (b) Unit cell. (c) Cross-section.

a full-wave EM simulation over a very large frequency bandwidth, i.e. DC to high GHz regimes, followed by a bit-by-bit time domain simulation with a large pseudorandom bit sequence (PRBS) to characterize the eye diagram at very low bit error rate (BER) contours.

In order to eliminate frequency sweeps over a large bandwidth from the optimization loop, we use the Efficient Bayesian Framework (EBF) as described in [86] and first build a parametric surrogate model of the frequency response of the channel. In EBF, frequency is considered as an additional input parameter. Hence, the training data required to build the model can be acquired by performing simulations at single frequency points, thereby

Table 6.3: Control Parameters of the High Speed Channel

Parameter		Unit	Min	Max
Line Width	l_w	μm	0.4	3
Line Thickness	t_c	μm	0.4	3
Line Spacing	l_s	μm	0.4	3
Substrate Height I	h_{sub1}	μm	1	5
Substrate Height II	h_{sub2}	μm	1	5
Substrate Height III	h_{sub3}	μm	1	5
P/G Grid Width	g_w	μm	10	40
P/G Grid Thickness	t_g	μm	0.4	3
Signal Misalignment Ratio	r_w		0	0.5
Frequency	f	GHz	0.05	20

greatly reducing the CPU time required to build the surrogate model. This model is then used in the optimization loop, along with a time-domain simulation, to directly optimize the eye opening. Note that although the surrogate model replaces the EM solver to characterize the frequency response, the time-domain simulation is still CPU intensive. The high-speed channel considered in this section is three single-ended embedded microstrip lines on a silicon interposer referenced to a non-ideal power/ground (P/G) grid as shown in Figure 6.6. The input parameters comprising a ten dimensional sample space are given in Table 6.3.

6.2.1 Channel Surrogate Model

In order to create the surrogate model of the microstrip channel, we use a GP with a SE kernel. Since the transmission lines considered in Figure 6.6(a) have discontinuities in its reference plane, the propagation constant varies along the direction of propagation. Hence, per unit length RLGC matrices can not be defined. We therefore consider the unit cell as in Figure 6.6(b) and model per unit cell (p.u.c) RLGC matrices of the microstrip channel.

First, 1000 samples based on uniform Latin Hypercube Sampling (LHS) are determined. Then, a full-wave EM solver, Ansys HFSS, is used to extract the p.u.c RLGC matrices. As we consider frequency as an input to the surrogate model, simulations are done at single frequency points rather than a sweep in the entire bandwidth. Here, each

set of geometrical parameters in the training data is evaluated at different frequency points. Hence, the total amount of CPU time required to collect training data is greatly reduced. Note that as any other geometrical parameter in Table 6.3, frequency points follow a uniform marginal distribution as determined by LHS. The dimensionality of the output space defined by the full rank p.u.c RLGC matrices is 36. Since the microstrip channel is a reciprocal network, it is sufficient to consider the elements in the lower triangular and the diagonal of the RLGC matrices, effectively reducing the number of outputs to 24.

Since the RLGC models are to be used in a time-domain simulation, predictions need to preserve stability of the interconnect model. For a lossy, linear time-invariant (LTI) system represented by RLGC elements, passivity is a sufficient condition stability [43]. For an RLGC network to be passive, each element in these matrices has to have a certain property. Diagonal terms of C & G matrices have to be non-negative and off-diagonal terms must be non-positive. Further, each element of R & L matrices has to be non-negative. These properties can be preserved by using a *log* transformation on the training data as a pre-processing step. This can then be reversed in the prediction stage by exponentiation of the predicted values, thereby ensuring every element in predicted RLGC matrices has the proper sign that ensures passivity and stability.

After the training data is collected and pre-processed using log transformation, 750 out of 1000 samples are standardized to have zero mean and unit standard deviation and used to perform the training of the GP model to maximize log marginal likelihood as explained in Chapter 2. Since GP models with SE kernel can only model a single output at a time, 24 independent GP models have been trained to predict p.u.c RLGC matrices. The remaining 250 samples are then used to assess the quality of the GP model using the normalized mean squared error (NMSE) criteria, given as:

$$NMSE = \frac{\sum_{i=1}^{N} \left(\hat{y}(x) - y(x)\right)^2}{\sum_{i=1}^{N} \left(y(x) - \frac{1}{N}\sum_{i=1}^{N} y(x)\right)^2} \tag{6.14}$$

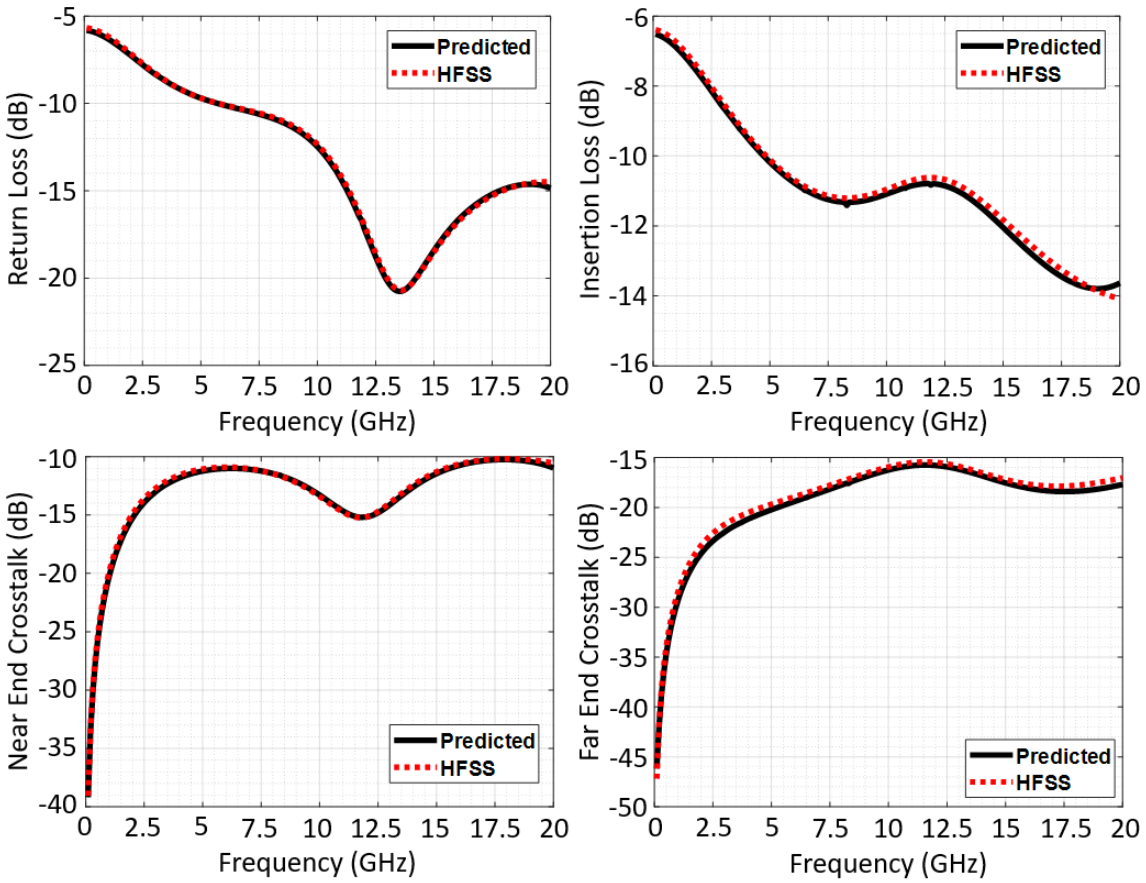


Figure 6.7: Comparison of the GP model with full-wave simulations for a channel of length 10 mm.

where $N = 250$; $y(x)$ is the validation data and $\hat{y}(x)$ represent predictions done by the GP model at validation samples.

As can be seen from Table 6.4, the NMSE for p.u.c RLGC matrices is kept less than 3.01%, showing the high quality of the predictive model. A further test for the model accuracy is performed by converting predicted RLGC matrices to S-Parameters of the unit cell. This is then cascaded to itself to form the frequency response of a channel with a length of 10 mm. For a random parameter assignment, the comparison for insertion loss (IL), return loss (RL), near-end (NEXT) and far-end (FEXT) crosstalk over the entire bandwidth can be seen in Figure 6.7.

Table 6.4: NMSE Values of the GP Model

Parameter	NMSE	Parameter	NMSE	Parameter	NMSE	Parameter	NMSE
R_{11}	0.0003	L_{11}	0.0212	G_{11}	0.0011	C_{11}	0.0041
R_{12}	0.0291	L_{12}	0.0009	G_{12}	0.0013	C_{12}	0.0079
R_{13}	0.0004	L_{13}	0.0075	G_{13}	0.0009	C_{13}	0.0023
R_{22}	0.0184	L_{22}	0.0003	G_{22}	0.0014	C_{22}	0.0057
R_{32}	0.0291	L_{32}	0.0012	G_{32}	0.0096	C_{32}	0.0030
R_{33}	0.0003	L_{33}	0.0301	G_{33}	0.0163	C_{33}	0.0049

6.2.2 Optimization of Interconnects

The objective function for the direct eye optimization is defined as:

$$f(x) = W_E H_E \tag{6.15}$$

where W_E and H_E denotes width and height of the eye diagram at a BER contour of 10^{-12}. The optimization loop starts by determining the geometrical parameters of the microstrip channel by the optimization algorithm, while fixing the P/G grid spacing, i.e. $g_s = 50\mu m$ in Figure 6.6(b), to fix the length of the unit cell. The p.u.c RLGC matrices over the entire frequency bandwidth are then determined by the GP model, converted to S-Parameters and then cascaded to each other to form the interconnect channel with a length of 10 mm. The channel S-Parameters are then used by a commercial circuit solver, Keysight ADS, to perform bit-by-bit simulation and generate the eye diagram at a data rate of 16 Gbps. The output impedance of the driver is fixed to 50 Ω and the load at the receiver is fixed as a 50 Ω resistor with a shunt capacitor of 1 pF to represent the pad parasitics. The eye width and height generated by ADS is then combined in Equation 6.15 and fed back into the optimization algorithm to proceed into next iteration.

The optimization results are summarized in Table 6.5 and in Figure 6.8. Optimization using DPT-BO resulted in an eye width and height of 53.13 ps and 0.54V along with a peak-to-peak jitter of 8.12 ps, corresponding to a improvement of 1.8%, 11.1%, 1.7% and

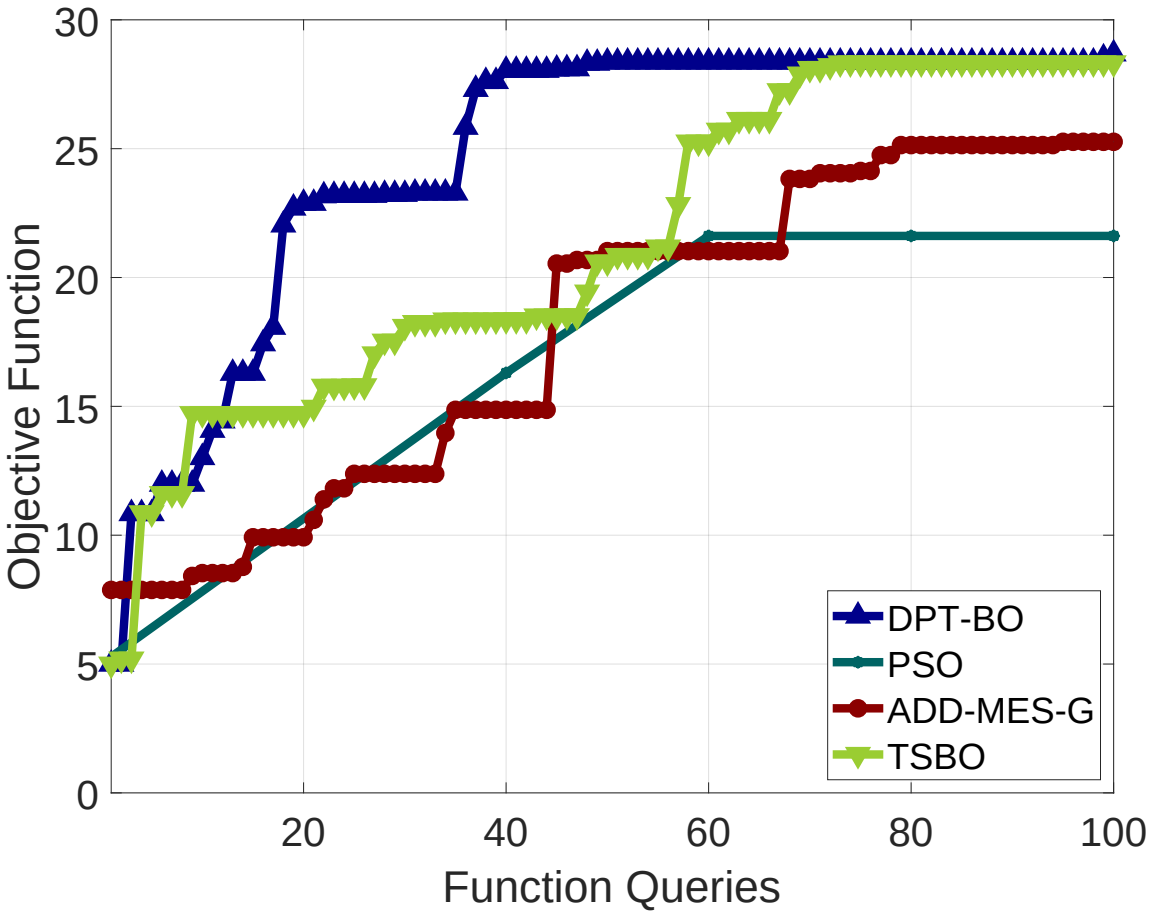


Figure 6.8: Performance of the proposed algorithm for maximizing the objective function in Equation 6.15.

Table 6.5: Optimization Results for High-Speed Channel

	PSO	**TSBO**	**ADD-MES-G**	**DPT-BO**
Eye Width (ps)	50.31	54.15	52.18	53.13
Eye Height (V)	0.46	0.52	0.48	0.54
Pk-Pk Jitter (ps)	10.3	8.04	8.26	8.12
AUC (Normalized)	1.48	1.19	1.41	1.00

5.3%, 14.8%, 21.2% as compared to ADD-MES-G and PSO, respectively. In addition, the normalized AUC values show that DPT-BO converged 1.41X and 1.48X faster as compared to ADD-MES-G and PSO, respectively. For the sake of completeness, the BO approach that uses the partitioning tree with only vertical expansion, TSBO in Chapter 3, is also applied to maximize the eye opening. As shown in Table 6.5, optimization using TSBO and DPT-BO have resulted in similar eye characteristics. However, DPT-BO converged 1.19X faster than TSBO. As explained in previous sections, only vertical expansion lacks proper exploitation of the sample space, hence, converges slowly as the problem dimensionality increases. This limits the use of TSBO in higher dimensional design problems considered in subsequent sections of this chapter.

6.3 Application 2: Sub-THz Substrate Integrated Waveguides

The second application chosen in this work is the optimization of a substrate integrated waveguide (SIW) with the goal of minimizing the losses over D-band. SIW technology is a very promising alternative to conventional microstrip and coplanar waveguide transmission lines as the operating frequency increases into the THz region. The limitation, however, is the increasing dielectric loss that tends to dominate the total transmission losses. To address this, SIWs designed on a multilayer PCB with an air cavity in the middle layer has attracted attention [87]. Optimization of such structures to minimize total transmission losses can be a very challenging problem given the non-linearity of the parameters.

In this section, we consider the SIW structure with air cavity inside a liquid crystal polymer (LCP) substrate ($\epsilon_r = 3.1, tan\delta = 0.005$ as shown in [88]) in Figure 6.9 and use DPT-BO to automatically find the optimal shape of the air cavity along with the geometrical parameters of SIW and microstrip to SIW transition. The input parameters comprise a 14 dimensional problem as shown in Table 6.6. Note that the length of the waveguide section is fixed to 2 mm. The air cavity is parameterized as two elliptical regions connected by a rectangular region as shown in Figure 6.9. We consider two cases namely, i) no air cavity

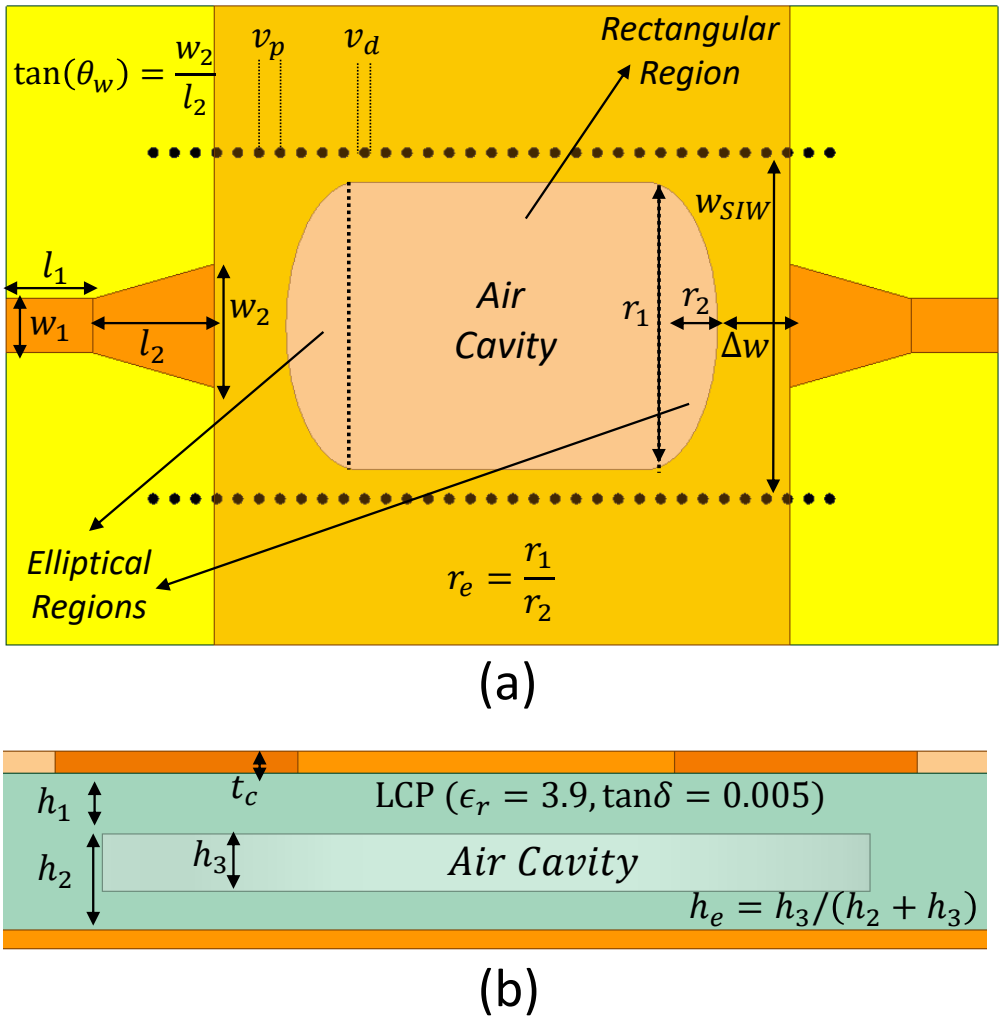


Figure 6.9: Structure of the SIW with elliptical air cavity. (a) Top view. (b) Cross-section.

and ii) air cavity. We compare the results with an all LCP structure that was designed through hand tuning and fabricated [88].

6.3.1 Optimization Setup

The objective function to be maximized is chosen as the minimum magnitude of the transmission coefficient over D-band, given as:

$$f(x) = \min(|S_{21}(f)|), \quad \text{where} \quad f \in [110, 170] \text{ GHz} \tag{6.16}$$

Table 6.6: Control Parameters of the SIW structure

Parameter		Unit	Min	Max
SIW Width	w_{SIW}	mm	0.5	2.5
Microstrip Width	w_1	mm	0.1	0.3
Microstrip Length	l_1	mm	0.2	1
Taper Angle	θ_w	deg	60	90
Taper Length	l_2	mm	0.2	1
Via Diameter	v_d	μm	20	60
Via Pitch	v_p	μm	50	150
Copper Thickness	t_c	μm	10	30
Substrate Height I	h_1	μm	10	30
Substrate Height II	h_2	μm	10	90
Air Cavity Major Radius	r_1	mm	0.1	10
Air Cavity Aspect Ratio	r_e	μm	0.1	10
Air Cavity Offset	Δ_w	mm	0.1	0.9
Air Cavity Ratio	h_e		0	1

We rely on EM simulation to account for the arbitrarily shaped air cavity and microstrip to SIW transition while capturing the effect of variations in all geometrical parameters. We use Ansys HFSS to extract the S-Parameters of the SIW to model the transmission losses in this section.

6.3.2 Results

The optimization results are summarized in Table 6.7 and Figure 6.10. Optimization using DPT-BO resulted in minimum $|S_{21}|$ value of -0.763 dB over D-band as compared to -1.082 dB and -1.163 dB using ADD-MES-G and PSO, corresponding to a 29.5% and 34.4% improvement in transmission losses, respectively. Though all algorithms started from the same initial point, normalized AUC values show that DPT-BO converged 1.35X and 1.42X faster as compared to ADD-MES-G and PSO, respectively.

To quantify the advantage of using an air cavity inside the LCP substrate, we use DPT-BO to optimize the SIW design without an air cavity and compare its performance with the design with the optimal air cavity as well as the performance of the hand-tuned SIW with

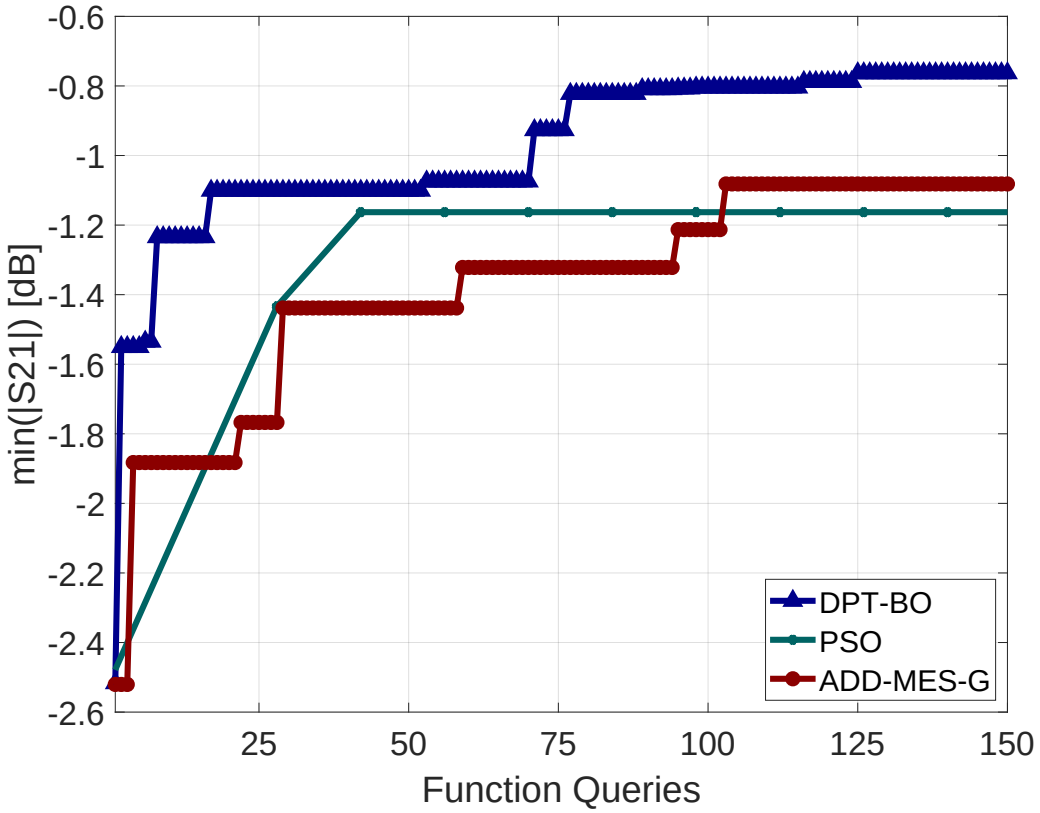


Figure 6.10: Performance of the proposed algorithm on loss minimization of SIW.

Table 6.7: Optimization Results for SIW

	PSO	**ADD-MES-G**	**DPT-BO**
Min. $\lvert S21(f)\rvert$ **(dB)**	-1.163	-1.082	-0.763
AUC (Normalized)	1.35	1.42	1.00

LCP in [88]. The minimum insertion loss for the optimal LCP SIW design is found to be -1.159 dB at 170 GHz and -1.911 dB for the hand-tuned design, corresponding to 34.2% and 60.1% worse performance compared to DPT-BO optimized design with the air cavity. This also shows that DPT-BO optimized LCP SIW has 39.4% better performance than the hand-tuned LCP SIW using the same structure.

Figure 6.12 shows that the optimal LCP only design has comparable performance with the designs optimized using both ADD-MES-G and PSO. This shows that both of these algorithms were stuck in a local maxima when maximizing the objective function in Equa-

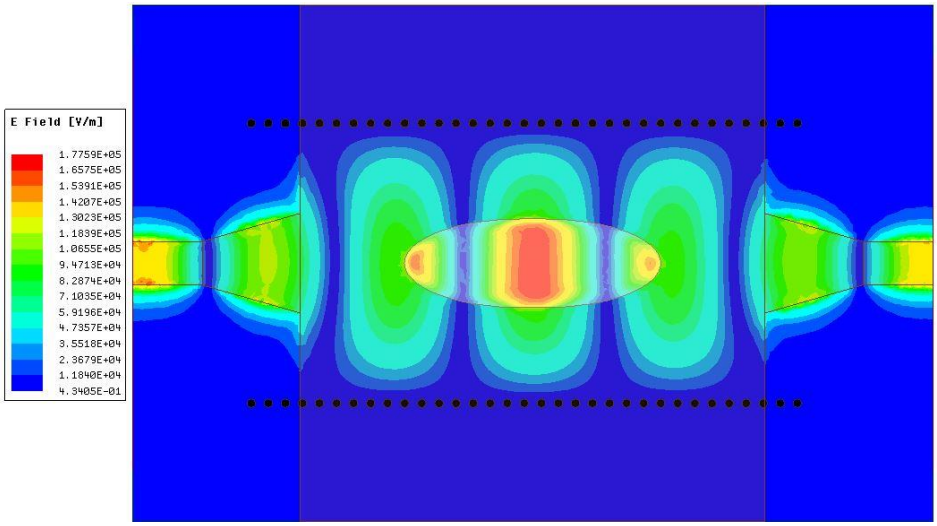
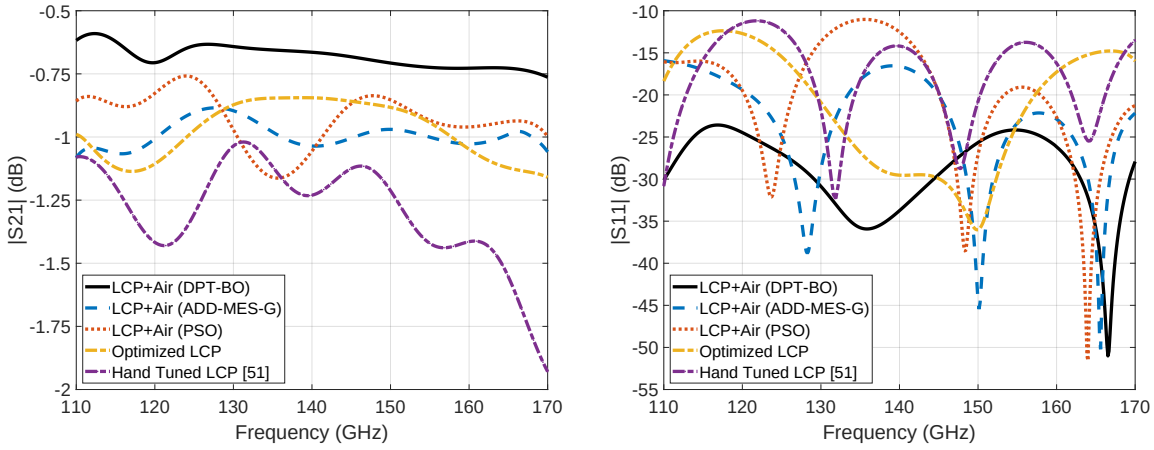


Figure 6.11: E-Field distribution at 170 GHz showing the air cavity of the optimized SIW.



(a) (b)

Figure 6.12: Performance of the optimized SIW compared to other designs.

tion 6.16, whereas DPT-BO was able get out of the local maxima and provide a better final design. Further, the E-Field distribution and $|S11|$ in Figure 6.11 and Figure 6.12b show that the reflections caused by the transition from LCP to air cavity are minimized to minimize the transmission loss. Cut-off frequencies for TE_{10} and TE_{20} modes of the design optimized by DPT-BO are simulated as $f_{c10} = 81.3$ GHz and $f_{c20} = 161.8$ GHz, showing losses due to dispersion and mode conversion are also minimized.

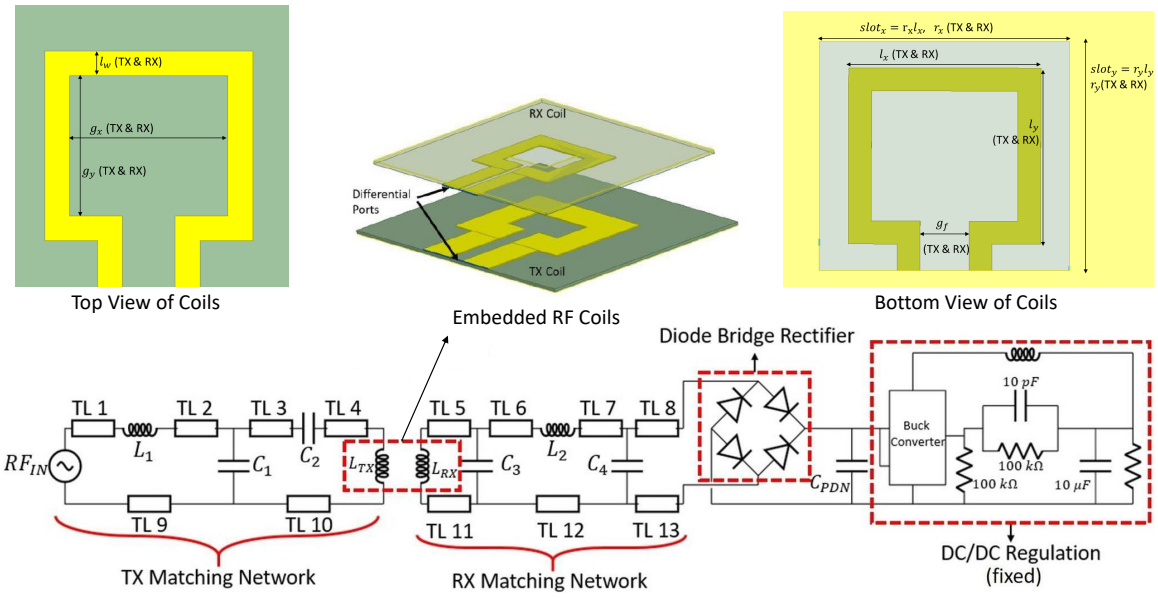


Figure 6.13: Geometry of the embedded RF coils defined by the control parameters. (a) WPT Structure. (b) Top view. (c) Bottom view

6.4 Application 3: Wireless Power Transfer based Power Delivery for IoT

The third application considered to evaluate the performance of the DPT-BO algorithm is optimization of an inductive coupling based wireless power transfer (WPT) system operating at 1 GHz. The architecture of the WPT system is given in Figure 6.13, which represents an integrated board solution, consisting of embedded rectangular RF coils as in Figure 6.13 connected to TX & RX matching networks, a full bridge diode rectifier and a buck converter (BC) for DC regulation [89]. The targeted application is efficient power delivery to IoT devices requiring input power in the range of 1-100 mW.

Two major design challenges in such power delivery architectures are to maximize conversion efficiency to reduce battery wastage and minimizing the area of the system to be mounted on the IoT device. In this section, we address the design complexity by formulating the design procedure as a system-level multi-objective optimization problem with the goal of maximizing the RF-DC conversion efficiency and minimizing the area of the RX coil. Previous work demonstrated a stage-by-stage optimization procedure for the same architecture considered in this chapter [90]. However, as common in RF systems,

Table 6.8: Control Parameters of WPT Architecture

Parameter		Unit	Min	Max
Inner Height of TX coil	$g_{y,TX}$	mm	1	5
Inner Height of RX coil	$g_{y,RX}$	mm	1	5
Inner Width of TX coil	$g_{x,TX}$	mm	1	5
Inner Width of RX coil	$g_{x,RX}$	mm	1	5
Line Width of TX coil	$l_{w,TX}$	mm	0.5	3
Line Width of RX coil	$l_{w,RX}$	mm	0.5	3
Feeding Gap for TX coil	$g_{f,TX}$	mm	0.5	3
Feeding Gap for RX coil	$g_{f,RX}$	mm	0.5	3
TX Vertical GND Cut-out Ratio	$slot_{y,TX}$		0.8	1.2
RX Vertical GND Cut-out Ratio	$slot_{y,RX}$		0.8	1.2
TX Horizontal GND Cut-out Ratio	$slot_{x,TX}$		0.8	1.2
RX Horizontal GND Cut-out Ratio	$slot_{x,RX}$		0.8	1.2
Capacitor I	C_1	pF	0.1	10
Capacitor II	C_2	pF	0.1	10
Capacitor III	C_3	pF	0.1	10
Capacitor IV	C_4	pF	0.1	10
Inductor I	L_1	nH	0.1	10
Inductor II	L_2	nH	0.1	10
Input Power	$P_{RF,IN}$	dBm	5	15
Widths of all TLINs (13 seperate parameters)	$w_{TL1,...,TL13}$	mil	15	45

individual blocks in the WPT architecture are coupled to each other and have contradicting design trade-offs, which raises the necessity to perform system-level optimization. This translates the optimization problem into a very a high-dimensional one along with a highly non-linear response surface with many local optima. The input parameters comprising a 32 dimensional sample space is given in Table 6.8.

6.4.1 Optimization Setup

The total system efficiency can be calculated by cascading the rectifier and DC-DC regulation efficiencies. Maximizing rectifier efficiency results in increased voltage on the load of the rectifier, i.e. input impedance of the BC. However, the BC requires lower input voltage to minimize switching losses. Since the design trade-off between RF-DC conversion and

DC-DC conversion stages is limited with this phenomena, we fix the DC regulation stage and focus on maximization of rectifier efficiency while minimizing input voltage of the BC. To minimize the overall size of the WPT architecture, lengths of all transmission lines are fixed to 0.5 mm based on process capabilities and the area of RX coil is minimized. Hence, the system-level optimization is formulated as maximizing the weighted sum of these objectives, given as:

$$f(x) = \sum_{i=1}^{3} w_i y_i \tag{6.17}$$

where y_1, y_2 and y_3 are rectifier efficiency, minimum input voltage of the buck converter and area of RX coil; $w_1 = 7$, $w_2 = -3.5$ and $w_3 = -3$ are the corresponding weights, chosen to prioritize efficiency over area. The BC considered in this section requires a minimum of 2.5V as input voltage [91]. Hence, y_2 is written as $|V_{\text{BC,IN}} - 2.5V|$ to set the target BC input voltage to 2.5V.

The simulation framework used starts with a full-wave EM simulation (Ansys HFSS) to extract the S-parameters of the RF coils that captures the impedance profile and the coupling between them. The S-Parameters are then fed into Keysight ADS to perform harmonic balance simulation to calculate the rectifier efficiency. The transmission distance between RF coils is fixed to 1 mm and the load of the buck converter, i.e. IoT device, is selected to operate at 1.2V with 7.2 mW input power. The simulation-measurement correlation for our framework can be found in [90].

6.4.2 Results

The optimization results are summarized in Figure 6.14 and Table 6.9. Optimization using DPT-BO resulted in 66.96% rectification efficiency, along with RX coil area of 11.04 mm^2 and BC input voltage of 2.72V, corresponding to 58.86% total system efficiency. Although electrical performance of the design found by ADD-MES-G is very similar to the one with DPT-BO, it provided a RX coil area of 19.26 mm^2, corresponding to 42.7% larger area as compared to DPT-BO. PSO provided a RX coil area of 7.48 mm^2, however, it per-

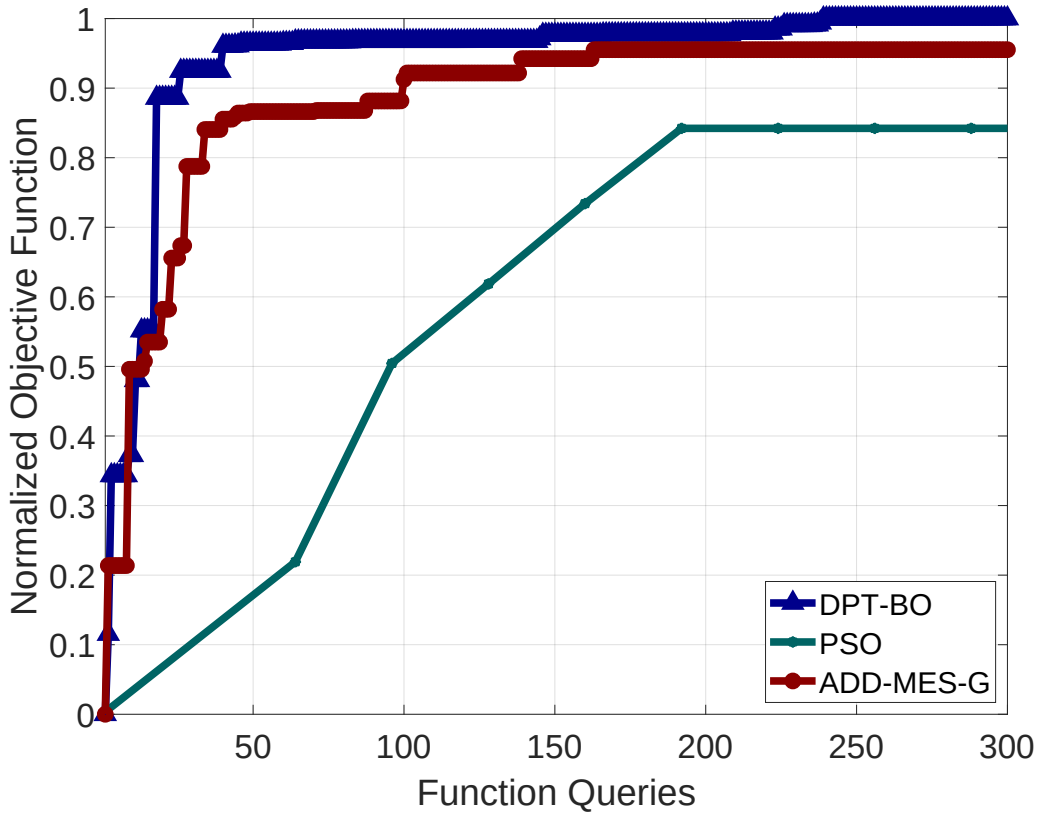


Figure 6.14: Performance comparison of the proposed algorithm on maximizing objective function in Equation 6.17.

formed significantly worse in terms of system efficiency with 13.03% and 13.74% reduced efficiency compared to ADD-MES-G and DPT-BO, respectively. The normalized AUC values show that DPT-BO converged 1.50X and 1.07X faster than PSO and ADD-MES-G, respectively.

In addition, Figure 6.15 shows the breakdown of the objective function in Equation 6.17 to its three components of rectifier efficiency, BC input voltage and RX coil area. This breakdown shows the design trade-offs made by each method. Both ADD-MES-G and DPT-BO have traded-off rectifier efficiency to reduce BC input voltage and minimize RX coil area, whereas PSO over-reduced the rectifier efficiency to minimize the area and got stuck in a local maxima.

Table 6.9: Optimization Results for WPT System

	PSO	ADD-MES-G	DPT-BO
RX Coil Area (mm^2)	7.48	19.26	11.04
Rectifier Efficiency (%)	53.25	65.72	66.96
Input Voltage of BC (V)	3.24	2.61	2.72
System Efficiency	45.83	58.86	59.57
AUC (Normalized)	1.50	1.07	1.00

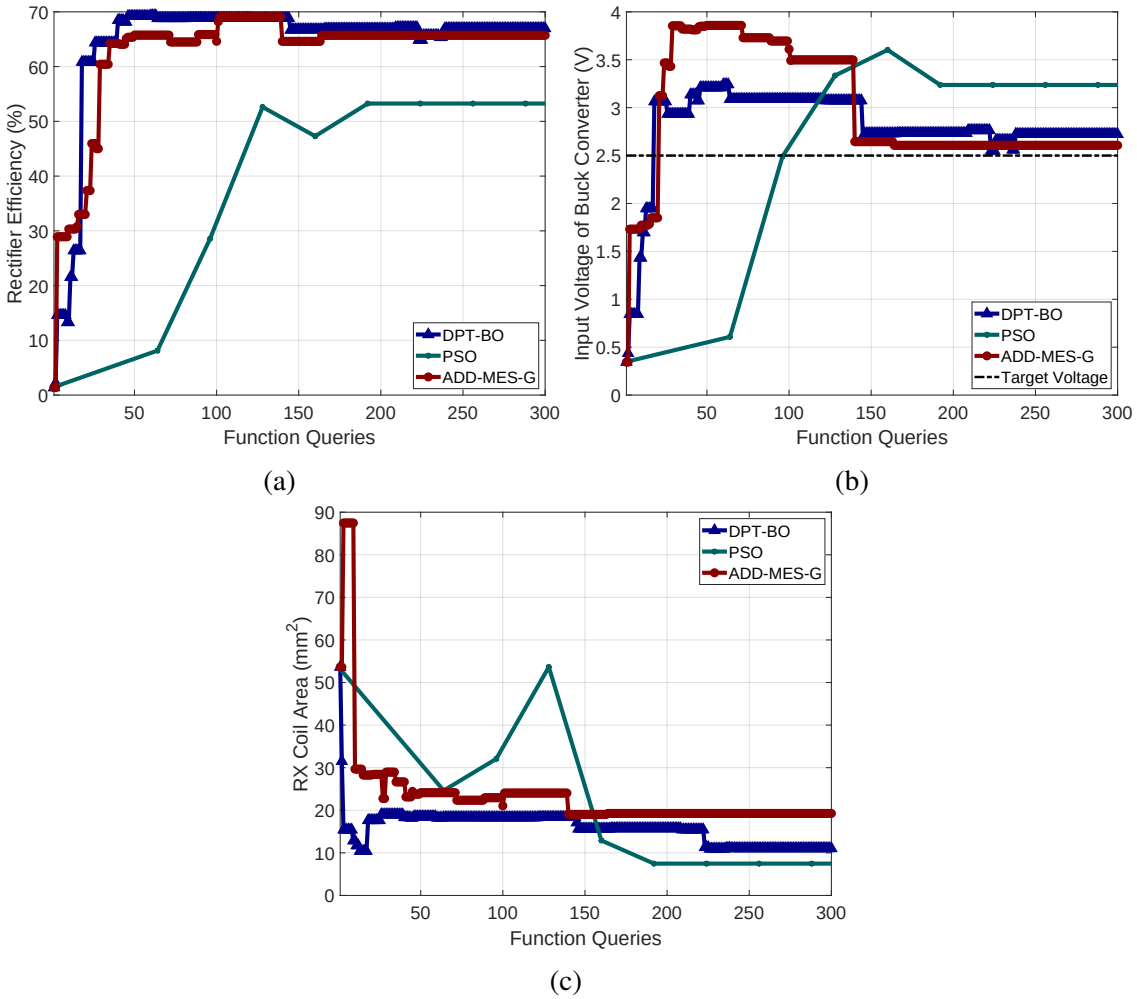


Figure 6.15: Performance of the optimized SIW compared to other designs.

6.5 Conclusion

In this chapter, we have proposed a new BO based global optimization method, *Bayesian Optimization with Deep Partitioning Tree*, that is designed for high dimensional problems. This is accomplished by utilizing an *Additive Gaussian Process* as the surrogate model. Unlike previous methods in the literature that uses ADD-GP with the disjoint group assumption, we have considered a fully additive decomposition to make DPT-BO better suited for high-frequency design optimization problems. In order to rapidly cover the high-dimensional sample space and eliminate auxiliary optimization, we have proposed and used a *Deep Partitioning Tree*. The proposed DPT-BO algorithm has been applied to a set of optimization test functions with different dimensionalities and response surface types. The proposed method showed better performance results with faster convergence.

DPT-BO has also been applied to three applications arising in high frequency electronic design, namely 1) 9 dimensional optimization of interconnects in high-speed channels to maximize eye diagram, 2) 14 dimensional optimization of a SIW structure with air cavity inside an LCP substrate operating in D-band and 3) 32 dimensional multi-objective optimization of a wireless power transfer based power delivery system to maximize the system efficiency. The results presented in this chapter shows the applicability of DPT-BO to perform global optimization on a variety of high-dimensional designs that operates in various frequency bands and therefore provides for a generic solution.

CHAPTER 7

MIXED-VARIABLE BAYESIAN OPTIMIZATION AND ITS APPLICATION TO POWER MODULE PACKAGE DESIGN

In Chapters 3 and 4 of this thesis, we have focused on Bayesian Optimization (BO) for hardware design optimization, where the design parameters were continuous variables or integer valued variables. In other words, the search domain $\mathbf{X}$ we considered was a D-dimensional hyper-rectangle or mixed-integer domain where some of the inputs can only take integer values. However, many packaging design problems involve optimization in a mixed-variable domain, where some of the parameters are continuous and some are categorical. Here, the categorical parameters refer to qualitative values and unlike integer-valued or continuous variables, it is either not possible or too complex to define a similarity metric between the different choices in the search domain. At a high-level, examples of categorical variables can be the color of an object where the choices are {"red", "green", "blue"} and the activation function in a neural network where the choices are {"tanh", "relu", "sigmoid"}. Since it is not possible to quantify the similarity between the choices as in continuous or discrete Euclidean domains, building a probabilistic model that can be utilized in the BO framework becomes very challenging.

In the field of packaging, mixed continuous and categorical variables are especially important in design of power electronics modules as used in electric vehicles. Design of such power module packages requires choosing between various cooling solutions, multi-layered packaging architectures, the optimal material combination to be used in each layer, and the physical placement of components such as diodes and switches. Each design choice needs to be evaluated through finite-element based multi-physics computer simulations to evaluate their thermal, mechanical and electrical performance. The multi-physics nature of the problem presents a challenge in defining a similarity metric between different material,

cooling and architecture-level choices since any two choices can have similar electrical performance but show a completely different thermal or mechanical behavior. Hence, it is more convenient to represent such design choices as categorical variables.

In this chapter, we develop a new BO technique to perform both single- and multi-objective design optimization in a mixed-variable domain consisting of multiple continuous and categorical variables. In particular, we develop a new Gaussian Process (GP) model that uses a learnable categorical embedding layer at the kernel level to learn an ordering between different categorical choices. We treat the elements of the embedding matrices as additional hyperparameters of the kernel function, thus, they are jointly learned along with other parameters through GP likelihood. This makes the proposed model a drop-in replacement of the conventional GPs used in the BO framework to extend into mixed-variable domains. We then provide sampling strategies for single- and multi-objective optimization settings and compare the results with existing approaches on synthetic test functions. Finally, we show the application of the proposed method on a comprehensive power module design problem where the goal is to co-optimize architecture, material, thickness and layout of an inverter package used in electric vehicles.

7.1 Mixed-Variable GP Model for Single-Objective Optimization

The conventional BO framework based on GPs can only handle input parameters that are continuous. This is due to the definition of the kernel function used to create the covariance matrix in Equation 2.7. The covariance matrix defines the pairwise similarity between any two input parameters and the kernel function solely determines the metric of similarity. For continuous parameters, this metric can be the Euclidean distance between the samples as in Matern 5/2 function in Equation 2.8 or other kernel choices given in Table 2.1. As it is not possible to represent qualitative categorical variables as numerical values, it is not possible to define such a similarity metric.

A commonly used method to address this issue is to represent each categorical variable

as a one-hot-encoded vector. Here, a categorical parameter that can take n possible values is represented by an $n \times 1$ sparse vector where all elements except one are zeros. The non-zero element then represents the active category. For instance, let $h = \{h^{(1)}, \cdots, h^{(n)}\}$ be a categorical variable that can take n possible values. The one-hot (OH) vector representation of h can then be written as

$$h_{OH}^{(i)} = e_i, \quad i = \{1, \cdots, n\} \tag{7.1}$$

where e_i is the standardized basis vector of size $n \times 1$ where the i^{th} element is equal to 1 and other elements are equal to zero. Although one-hot representation is used in some BO strategies along with conventional kernel functions [92], there are two main limitations. The conventional GP kernels are differentiable with respect to the input space, meaning that the scope of the functions that the GP model can replicate are also differentiable. Since the one-hot vector is non-differentiable, this significantly limits the scope of functions that can be predicted by the GP, thereby significantly degrading the convergence performance if used for BO purposes. The second limitation is the significantly increased dimensionality, which makes it computationally more challenging to train the GP and optimize the acquisition function. This computational challenge is exacerbated when there are more than one categorical variables to be optimized since each variable will have its own one-hot representation.

7.1.1 Model Structure

To address these limitations of the one-hot encoding approach, we propose to use a kernel-level categorical embedding approach where we learn a non-linear projection to convert the high-dimensional, non-differentiable one-hot encoded space into a low-dimensional and continuous latent space. Since the learned latent space is differentiable, a conventional kernel function such as Matern 3/2 can then be defined over this latent space.

Let the optimization problem contain a total of $\mathcal{C}$ categorical parameters, each having n_c possible choices. To learn the low-dimensional latent space, we first represent the original mixed-variable space as

$$\mathbf{x} = [\mathbf{x}_D, \mathbf{h}_\mathcal{C}] \tag{7.2}$$

where

$$\mathbf{h}_\mathcal{C} = \left[\mathbf{h}_{(1,OH)}, \cdots, \mathbf{h}_{(\mathcal{C},OH)}\right], \tag{7.3}$$

$\mathbf{x}_D$ is the D-dimensional continuous variable vector and $\mathbf{h}_{(c,OH)}$ is the n_c-dimensional one-hot representation of the c^{th} categorical variable as in Equation 7.1. The total dimensionality of $\mathbf{x}$ in Equation 7.2 then equals to $D_x = D + \sum_c^\mathcal{C} n_c$. We then define the kernel function for the mixed-variable space as

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') =& k\left([\mathbf{x}_D, \mathbf{h}_\mathcal{C}], [\mathbf{x}'_D, \mathbf{h}'_\mathcal{C}]\right) \\ =& k\left([\mathbf{x}_D, g(\mathbf{W}_\mathcal{C}\mathbf{h}_\mathcal{C})], [\mathbf{x}'_D, g(\mathbf{W}_\mathcal{C}\mathbf{h}'_\mathcal{C})]\right) \\ =& k\left([\mathbf{x}_D, \mathbf{Z}_\mathcal{C}], [\mathbf{x}'_D, \mathbf{Z}'_\mathcal{C}]\right) \end{aligned} \tag{7.4}$$

where

$$\mathbf{W}_\mathcal{C} = \{\mathbf{W}_1^\mathcal{C}, \cdots, \mathbf{W}_\mathcal{C}^\mathcal{C}\} \tag{7.5}$$

is the collection of linear embedding matrices, each having a size of $n_{z,c} \times n_c$. In Equation 7.4, each embedding matrix converts the categorical variables into their respective $n_{z,c}$-dimensional, continuous latent space $\mathbf{z}_c = \mathbf{g}(W_c^\mathcal{C}\mathbf{h}_{c,OH})$ and the domain that represents the collection of latent spaces for each categorical variable is denoted as $\mathbf{Z}_\mathcal{C} = [\mathbf{z}_1, \cdots, \mathbf{z}_\mathcal{C}]$. The function $g(z) = tanh(z)$ is introduced here to bound the domain of $\mathbf{Z}_\mathcal{C}$ to $[-1, 1]$ and introduce non-stationary behavior to the GP model. It should be noted here that since all the transformation operations in Equation 7.4 are performed inside the kernel function of the GP, it corresponds to an input-warping operation as in [93] and allows the kernel in Equation 7.4 to be used as a drop-in replacement of the conventional kernels. This ef-

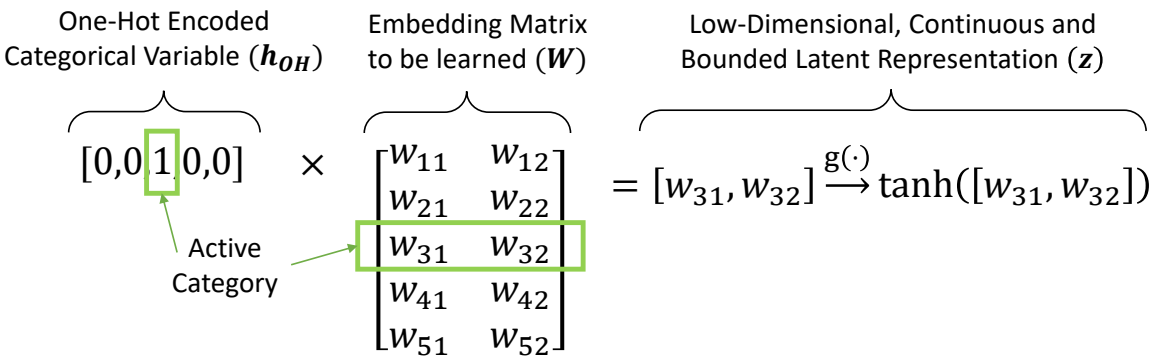


Figure 7.1: Illustration of the embedding operation to convert a categorical variable to a continuous domain. The dimensionality of the latent space is chosen as $ceil(n_c/2)$.

fectively enables extending any GP-based BO framework to a mixed-variable setting by simply changing the kernel function. This embedding operation in Equation 7.4 is further illustrated in Figure 7.1. It should also be noted that the latent domain dimensionality, $n_{z,c}$, is a hyperparameter that needs to be chosen apriori. In our experiments, we found that $n_{z,c} = ceil(n_c/2)$ provides a good performance, hence, we use this value for numerical examples in the next sections.

Including $\mathbf{W}_{\mathcal{C}}$ as part of the kernel function means that all the elements of each embedding matrix to become hyperparameters of the overall mixed-variable GP model. In order to learn a descriptive latent space that best describes the data at hand, we need to jointly learn all these hyperparameters through the GP likelihood as in Chapter 2. For convenience, we repeat the log-likelihood function of the GP that will be used to learn these hyperparameters here:

$$\mathcal{L} = -\frac{1}{2}\mathbf{Y}_t^T K_{\mathbf{X}_t}^{-1}\mathbf{Y}_t - \frac{1}{2}\log|K_{\mathbf{X}_t}| \tag{7.6}$$

where $|\cdot|$ is the matrix determinant operator. The learning of the embedding matrices can be done using regular gradient-descent based methods that are used to train the GP by using the chain-rule of derivatives [94]. Let $\gamma = \{\mathbf{W}_{\mathcal{C}}, \theta\}$ denote the hyperparameters of the mixed-variable GP-model where $W_{\mathcal{C}}$ are the elements of the embedding matrices and θ is the hyperparameters of the kernel function, chosen as Matern 3/2 given in Table 2.1. If

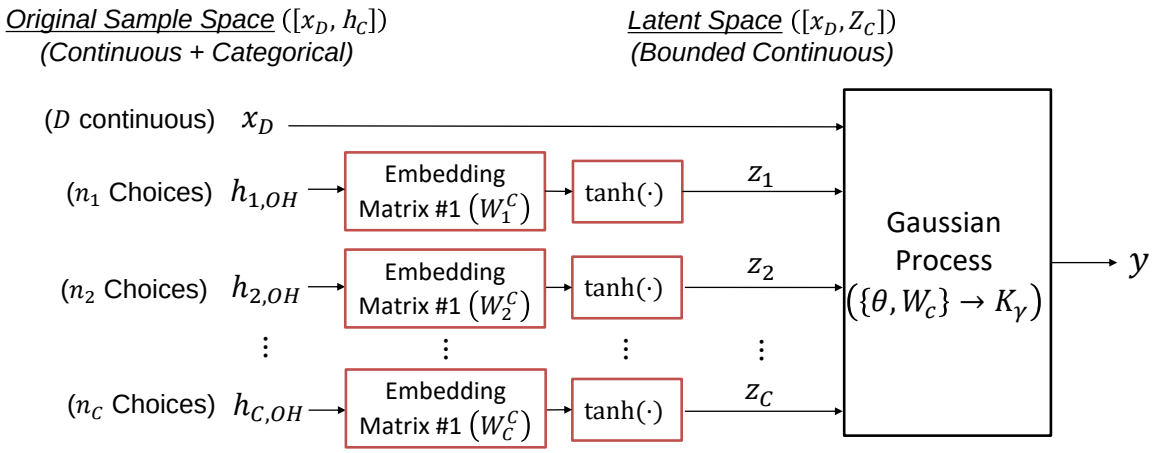


Figure 7.2: The structure of the proposed mixed categorical and continuous variable GP model.

we denote the covariance matrix calculated using γ as K_γ, the gradients can be written as

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \theta} &= \frac{\partial \mathcal{L}}{\partial K_\gamma} \frac{\partial K_\gamma}{\partial \theta} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{W}_C} &= \frac{\partial \mathcal{L}}{\partial K_\gamma} \frac{\partial K_\gamma}{\partial g(\mathbf{W}_C \mathbf{h}_C)} \frac{\partial g(\mathbf{W}_C \mathbf{h}_C)}{\partial \mathbf{W}_C} \end{aligned} \tag{7.7}$$

where $\mathcal{L}$ is the GP likelihood as in Equation 7.6. The overall structure of the proposed mixed-variable GP model is given in Figure 7.2.

Once the GP model is trained, we still need to optimize the acquisition function, $u(x)$, chosen as UCB as in Equation 2.20, to select the next sampling point, x_{t+1}. As this auxiliary optimization needs to be done in the mixed-variable space, gradient-based methods can not be utilized here. Since $u(x)$ is very fast-to-evaluate, there are multiple ways to perform this auxiliary optimization such as sampling based random search, per-category optimization or evolutionary algorithms. In this chapter, we use a mixed-integer based genetic algorithm (GA) [95] to optimize $u(x)$ in the original mixed-variable domain and obtain x_{t+1}. Once this is done, we perform a function query to get y_{t+1} and proceed into the next iteration.

7.1.2 Experiments on Synthetic Functions

In order to test the performance of the proposed method, we have considered four test functions with a different number of continuous and categorical variables. Here, we have modified some common benchmarking problems used to evaluate BO methods in the continuous domain by introducing categorical variables using different schemes. The functions we considered can be listed as:

Branin-2D2C contains 2 continuous variables ($D = 2$) and 2 categorical variables ($\mathcal{C} = 2$). Each categorical variable has 10 choices where each level corresponds to an integer between 0 and 9, i.e., $h_i \in [0, 1, \cdots, 9]$. The function can be written as

$$y = f(x_D)S_1 + S_2 \tag{7.8}$$

where $f(x_D)$ is the 2-D Branin-Hoo function [68], $S_1 = -1 + 0.25h_1$ and $S_2 = -1 + 0.25h_2$.

Hart-6D3C contains 6 continuous variables ($D = 6$) and 3 categorical variables ($\mathcal{C} = 3$). Each categorical variable has 5 choices where each level corresponds to an integer between 0 and 4, i.e., $h_i \in [0, 1, \cdots, 4]$. The function can be written as

$$y = f(x_D)S_1 + S_2 + S_3 \tag{7.9}$$

where $f(x_D)$ is the 6-D Hartmann function [68],

$$S_1 = 1 - (0.25h_1 - 0.5)^2, \;\; S_2 = \frac{2 + \tanh(0.5h_2 - 1)}{3}$$

and $S_3 = -1 + 0.25h_3$.

Ackley-d5C is a function class that contains $D = d$ continuous variables and 5 categorical variables. Each categorical variable has 5 choices where each level corresponds to an

integer between 0 and 4, i.e., $h_i \in [0, 1, \cdots, 4]$. The function is calculated as:

$$y = f(x_n)$$

where

$$x_n = [x_1, \cdots, x_d, z_1, \cdots, z_5], \;\; z_i = -1 + h_i$$

and $f(x_n)$ is the $(d + 5)$ dimensional Ackley function [68].

We compare the performance of the proposed method with two methods, namely One-Hot BO [96] and CoCaBO [97]. One-Hot BO is the aforementioned commonly used BO technique to handle categorical variables. Here, categorical variables are first converted into a high-dimensional one-hot representation. Then, a regular BO framework is applied to this high-dimensional space where one-hot vectors are treated as continuous variables and rounded to nearest category just before evaluating the objective function. CoCaBO is a very recent method that mixes multi-armed bandits with GPs to extend BO to mixed-variables spaces. Performance evaluation criteria in these experiments are: 1) minimum value found after a pre-determined number of function evaluations (y_{min}) and 2) normalized area under curve (AUC) as given in Equation 6.13. The normalization here is with respect to the AUC of the fastest algorithm on a given problem. We repeated each experiment five times, each time taking five samples based on Latin Hypercube Sampling (LHS) to act as initial samples.

Results of these experiments are provided Table 7.1 and Figure 7.3. It can be seen that the proposed method outperforms both One-Hot BO and CoCaBO in terms of both y_{min} and AUC in all four of the test problems. Furthermore, AUC values in Table 7.1 show that when the number of total categorical combinations increases, the performance gap between the proposed method and other algorithms start to increase. This shows the effectiveness of the proposed mixed-variable GP model in discovering the underlying structure of the categorical domains even when the number of total categorical combinations tends into the

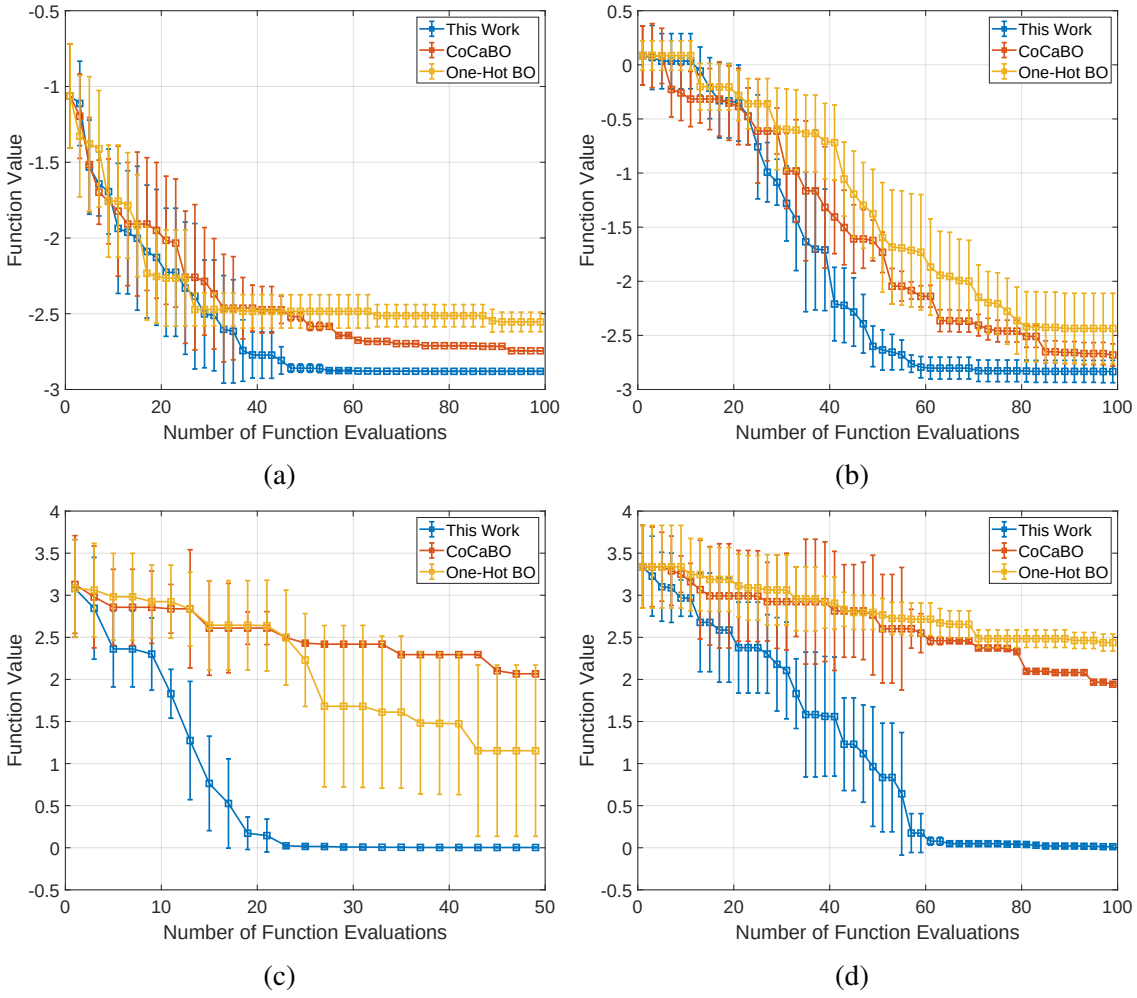


Figure 7.3: Performance of the proposed method on synthetic functions. (a) Branin-2D2C. (b) Hart-6D3C. (c) Ackley-1D5C. (d) Ackley-5D5C. The solid curves and error bars represent the mean and standard deviation of five random initializations, respectively.

Table 7.1: Minimum values found by each algorithm and their corresponding AUC on minimizing synthetic test functions.

	One-Hot BO		**CoCaBO**		**This Work**	
	y_{min}	AUC	y_{min}	AUC	y_{min}	AUC
Branin-2D2C	-2.55	1.10	-2.74	1.07	-2.88	1.00
Hart-6D3C	-2.44	1.34	-2.68	1.19	-2.83	1.00
Ackley-1D5C	0.94	2.39	2.05	3.74	0.00	1.00
Ackley-5D5C	2.43	2.43	1.94	2.29	0.01	1.00

thousands region.

7.2 Mixed-Variable Multi-Objective Optimization

Although single-objective BO has many successful uses, many of the packaging design problems deal with multi-objective optimization of multi-physics structures. Without loss of generality, in a mixed-variable domain, the problem to be solved can be written as

$$\widetilde{\mathbf{x}} = \underset{\mathbf{x}=[\mathbf{x}_D,\mathbf{h}_C]\in\mathbf{X}}{\arg\min} \big(f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_M(\mathbf{x})\big) \tag{7.10}$$

where $(f_1, \cdots, f_M)$ are blackbox functions to be minimized simultaneously. Most commonly, a single $\widetilde{\mathbf{x}}$ that simultaneously minimizes all the objectives does not exist since the multiple-objectives typically have certain trade-offs. As a result, the desired outcome is a set of non-dominated solutions, i.e., Pareto front, where the performance of each solution with respect to one objective can not be improved without degrading its performance on other objectives.

In this section, we extend the proposed mixed-variable BO method in previous section to the multi-objective setting where the goal is to estimate the Pareto Front of multiple blackbox functions. For brevity, we refer readers to [98] for a background on multi-objective optimization and Pareto front.

7.2.1 Model Structure

Multi-objective BO in the continuous domain is a widely studied area that resulted in several widely used approaches. Two main model structures have been adopted in the literature of multi-objective BO, namely using a single GP and multiple GPs. In the former, multiple objectives are combined using a weighting vector that is sampled randomly at each iteration of BO. A single GP is then trained on the weighted sum of multiple-objectives, which reduces the problem into a single-objective BO where acquisition function strategies devel-

oped for single-objective optimization can be directly used [99]. In the latter, a separate GP is used for each objective and the next sampling point is determined using an acquisition function that combines posterior mean and variances of multiple GPs [100]. The random weight sampling strategy is an unreliable approach since it does encourage or guarantee the final set of non-dominated solutions to be spread from each other. The spread of the solutions is a highly desired outcome of multi-objective optimization since it allows to analyze the trade-offs involved in the problem.

Since trade-off analysis is highly important in engineering problems, we adapt the multiple-GP approach in this chapter. Specifically, at the T^{th} iteration of the BO, let $(\mathbf{X}_T, \mathbf{Y}_T^M)$ be the data collected where $\mathbf{Y}_t^M = \{\mathbf{y}_T^1, \cdots, \mathbf{y}_T^M\}$. We then form M independent mixed-variable GPs as

$$\mathcal{GP}^i = \mathcal{N}(\mu^{(i)}(X_T), K_{\gamma^i}), \;\; \forall i \in \{1, \cdots, M\} \tag{7.11}$$

where γ^i is hyperparameter vector of the i^{th} GP model and is a collection of the elements of embedding matrices and kernel parameters, i.e., $\gamma^i = \{\theta^i, \mathbf{W}_{\mathcal{C}}^i\}$. Note that defining a separate hyperparameter vector for each GP corresponds to a separate set of embedding matrices for each objective. Since categorical variables can behave differently on different objectives, defining a separate set of embedding matrices corresponds to learning a separate latent space that best describes the corresponding objective function.

Once M independent GP models are defined, they can be trained in an efficient manner using a single gradient-descent loop over sum of their negative log-likelihoods. This can be written as

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \cdots + \mathcal{L}_M \tag{7.12}$$

$$\frac{\partial \mathcal{L}}{\partial(\gamma^1, \cdots, \gamma^M)} = \frac{\partial \mathcal{L}_1}{\partial \gamma^1} + \cdots + \frac{\partial \mathcal{L}_M}{\partial \gamma^M} \tag{7.13}$$

where $\mathcal{L}_i$ is the negative log-likelihood of the i^{th} GP model. The overall structure of the

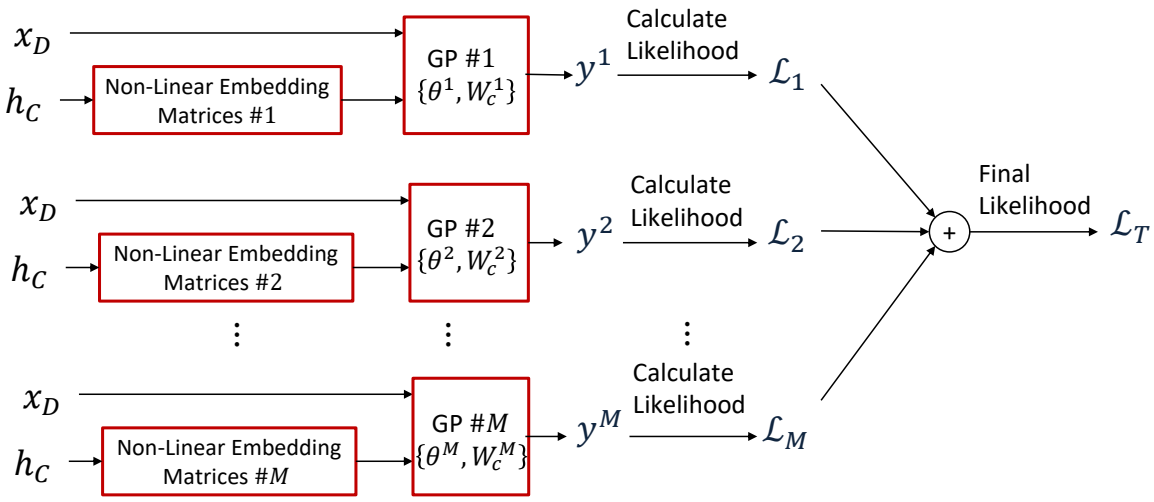


Figure 7.4: Extension of the proposed mixed-variable GP model for multi-objective BO. A separate set of embedding matrices for each objective allows to learn a unique latent domain for each objective.

mixed-variable GP model for multi-objective BO is given in Figure 7.4.

7.2.2 Sampling Strategy

Similar to the single-objective case, the proposed mixed-variable GP model for multi-objective BO allows to use existing sampling strategies developed for continuous variables. Popular strategies here employ a hypervolume based metric such as hypervolume-based probability of improvement (HV-PoI) and expected improvement (HV-EI) [101]. Here, posterior means and variances of multiple GPs are combined in a single-objective acquisition function where a single-objective auxiliary optimization is used to select the next sampling point. However, hypervolume based acquisition functions require numerical integration in the objective space, i.e., M-dimensional integration, that makes auxiliary optimization procedure computationally infeasible for $M > 2$ [102]. In addition, reducing the acquisition function to a single-objective problem has a known drawback of over-exploitation that results in sub-optimal solutions [103, 104].

To address the computational bottleneck of optimizing the acquisition function for $M > 2$, we propose an efficient sampling strategy that can be utilized for both continuous and

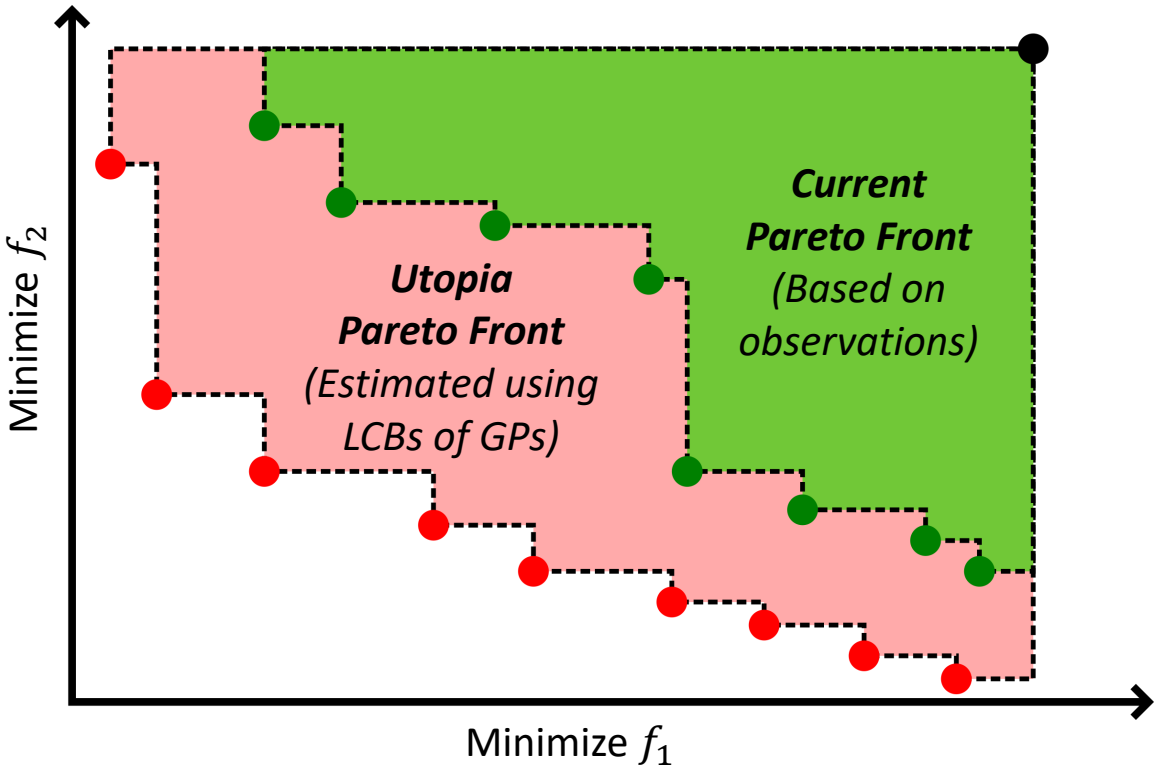


Figure 7.5: Proposed sampling strategy for multi-objective BO. The utopia set is the Pareto front of LCBs of multiple GPs. The next sample is the point in the utopia set such that if added to the current set, the increment over current hypervolume is the maximum.

mixed-variable BO setting. We first find the non-dominated solutions among the currently observed points and denote this set P_T and hypervolume of P_T as HV_P. Note that P_T contains $p \leq T$ solutions since not all the T observations can be non-dominated. Then, we calculate the Pareto front of lower confidence bounds (LCB) of M independent GPs. That is, we use a multi-objective evolutionary algorithm to get a set of non-dominated solutions, U_T, that act as our estimate of Pareto front of LCBs. This can be written as

$$U_T = \text{ParetoFront}\left(LCB^1, LCB^2, \cdots, LCB^M\right) \tag{7.14}$$

where

$$LCB^i = \mu^i(\mathbf{x}) - K\sigma^i(\mathbf{x}), \;\; \forall i = \{1, \cdots, M\} \tag{7.15}$$

and μ^i and σ^i are posterior mean and standard deviation of $\mathcal{GP}^i$. We call U_T as the *utopia*

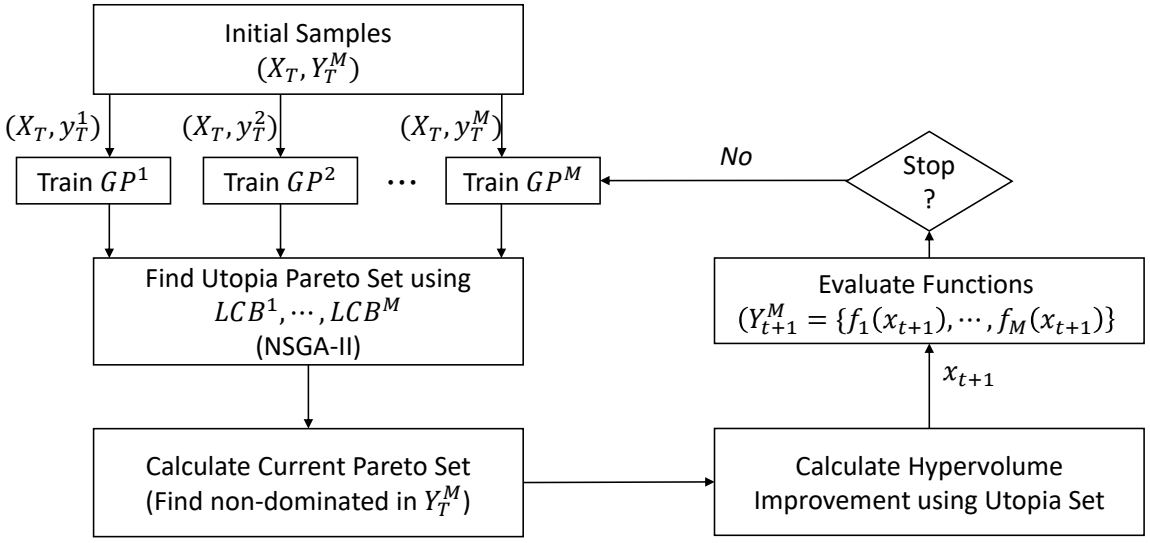


Figure 7.6: Block diagram of the proposed multi-objective mixed-variable BO method.

set as the set U_T is the most optimistic, i.e., best possible but not likely to be achieved, Pareto front estimate that can be obtained based on LCBs of M independent GPs. It should be noted here that since σ^i in Equation 7.15 is always non-negative, the solutions in U_T dominate the solutions in P_T as illustrated in Figure 7.5.

After U_T is obtained, we need to select one of the solutions among it and set the next sampling point to its corresponding input vector. Here, we calculate which solution in U_T, if added to P_T, would provide the highest improvement over the hypervolume of the current set, HV_P. More specifically, let u be a solution in the set U_T and $\widehat{x}_u$ be its corresponding input vector. The hypervolume improvement can then be calculated as

$$\Delta HV = HV\left(\{P_T, u\}\right) - HV_P, \;\; \forall u \in U_T \tag{7.16}$$

where $HV\left(\{P_T, u\}\right)$ is the hypervolume of the union of P_T and u after discarding dominated solutions. The next sampling point is then the input vector that corresponds to u that provides the largest ΔHV. The overall sampling strategy is summarized in Figure 7.6.

7.2.3 Experiments on Synthetic Functions

We test the performance of the proposed method on three test functions with different number of continuous & categorical variables and number of objectives. Similar to the single-objective test functions, we have modified common problems used to evaluate BO methods in continuous domains and introduced categorical variables. The functions we considered can be listed as:

OKA1-2D2C contains 2 continuous variables ($D = 2$) and 2 categorical variables ($\mathcal{C} = 2$). Each categorical variable has 10 choices where each level corresponds to an integer between 0 and 9, i.e. $h_i \in [0, 1, \cdots, 9]$. There are two objectives to be minimized, which can be written as

$$y_1 = (S_1 f_1(x_D) + S_2 + 3)/8 \tag{7.17}$$

$$y_2 = -S_1 f_2(x_D) - S_2 \tag{7.18}$$

where $f_1(x_D)$ and $f_2(x_D)$ are the 2-D OKA1 function with 2 objectives [99], $S_1 = -0.5 + 0.25h_1$ and $S_2 = -0.5 + 0.25h_2$.

DTLZ1a-1D5C contains 1 continuous variable ($D = 1$) and 5 categorical variables ($\mathcal{C} = 5$). Each categorical variable has 5 choices where each level corresponds to an integer between 0 and 4, i.e. $h_i \in [0, 1, \cdots, 4]$. There are two objectives to be minimized, which can be written as

$$y_1 = f_1(x_n), \quad y_2 = f_2(x_n) \tag{7.19}$$

where

$$x_n = [x_1, z_1, z_2, z_3, z_4, z_5], \;\; z_i = 0.25h_i$$

and $f_1(x_n)$ and $f_2(x_n)$ are the 6-dimensional DTLZ1a function with 2 objectives [99].

VLMOP3-2D5C contains 2 continuous variable ($D = 2$) and 5 categorical variables ($\mathcal{C} =$

5). Each categorical variable has 5 choices where each level corresponds to an integer between 0 and 4, i.e. $h_i \in [0, 1, \cdots, 4]$. There are three objectives to be minimized, which can be written as

$$y_1 = (S_1 f_1(x_D) + S_2 + S_3^2 + S_4^3 + 1)/10 \tag{7.20}$$

$$y_2 = (-S_1 f_2(x_D) - S_2 + S_3^2 - S_4^3 + 60)/50 \tag{7.21}$$

$$y_3 = (S_1 f_3(x_D) + S_2 + S_3^2 - S_4^3 + 0.5)/2 \tag{7.22}$$

where

$$S_1 = 1 - (0.25h_1 - 0.5)^2, \;\; S_2 = \frac{2 + \tanh(0.25h_2 - 0.5)}{3},$$

$S_i = -0.5 + 0.25h_i$ for $i = \{3, 4, 5\}$ and $f_1(x_D)$, $f_2(x_D)$, $f_3(x_D)$ are the 2-D VLMOP3 function with 3 objectives [99].

We compare the performance of the proposed method with a recently developed multi-objective BO method that is built into the open source library named Dragonfly [105] and an evolutionary algorithm, NSGA-II. Since Dragonfly is built for continuous domain optimization, we use one-hot encoding to handle categorical variables. Performance evaluation criteria we use for multi-objective optimization are the hypervolume indicator (HV) and AUC to measure rate of convergence for different methods as in previous section. We repeated each experiment five times, each time taking 20 initial samples based on LHS.

The results of these experiments are summarized in Table 7.2 and Figure 7.7. The difference between HV indicators show that the proposed method significantly outperforms existing methods in terms of the quality of the final Pareto set obtained. In addition, the normalized AUC values show that the proposed method converges 1.27X-3.56X and 1.35-3.09X faster as compared to NSGA-II and Dragonfly, respectively.

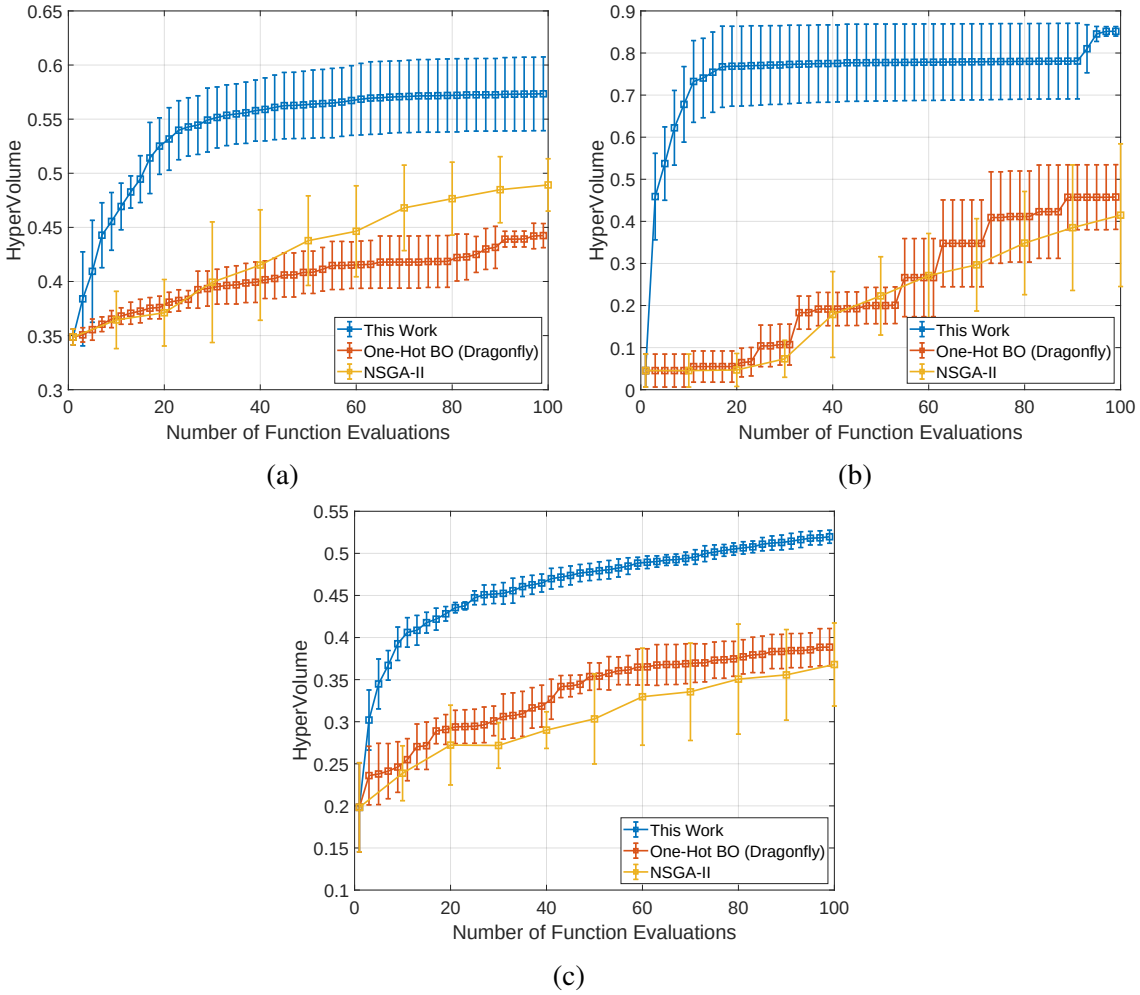


Figure 7.7: Performance of the proposed method on synthetic functions. (a) OKA1-2D2C. (b) DTLZ1a-1D5C. (c) VLMOP3-2D5C. The solid curves and error bars represent the mean and standard deviation of five random initializations, respectively.

Table 7.2: Hypervolumes of Pareto fronts estimated by each algorithm and their corresponding AUC on multi-objective test functions.

	NSGA-II		**Dragonfly**		**This Work**	
	HV	AUC	HV	AUC	HV	AUC
OKA1-2D2C	0.49	1.27	0.44	1.35	0.57	1.00
DTLZ1a-1D5C	0.41	3.56	0.46	3.09	0.85	1.00
VLMOP3-2D5C	0.37	1.54	0.38	1.39	0.52	1.00

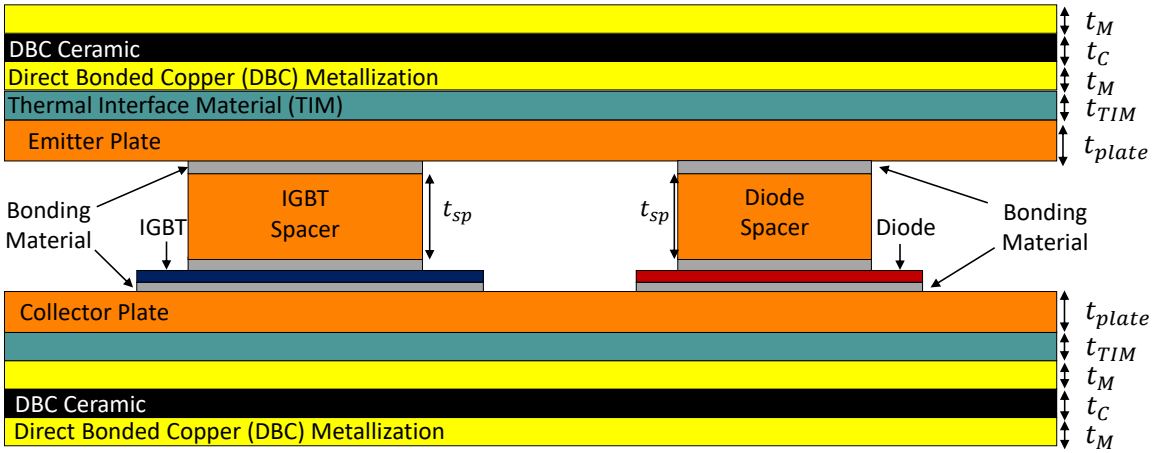


Figure 7.8: Single-cell cross-section of the DENSO power module.

7.3 Application to Power Module Design for Automotive Packaging

Reducing the volume of modern power electronics module packages used in electric vehicles is a challenging design problem. Design of such power module packages requires handling multiple trade-offs regarding volume, electrical parasitics, device temperatures and other performance criteria. The design problem involves determining the correct multi-layered packaging architecture and material types to be used for each layer of the package, choosing an appropriate cooling solution to ensure device temperatures remain below operational limits and determining the correct physical layout to minimize electrical parasitics and manage heat dissipation.

In this section, we formulate the design of a power module package as a mixed-variable, multi-objective and multi-physics optimization problem and use the method presented in Section IV. In particular, we consider the design of a half-bridge inverter power card package [106] and aim to find the Pareto front of 4 objectives, namely parasitic inductance (L), capacitance (C), maximum junction temperature (T_{max}) and package volume (V). For brevity, we refer readers to [107] for the definitions of parasitic L and C used in this chapter.

The thickness and layout of the package is determined by 8 continuous variables as illustrated in Figure 7.8 and Figure 7.9. Material and structural choices related to the de-

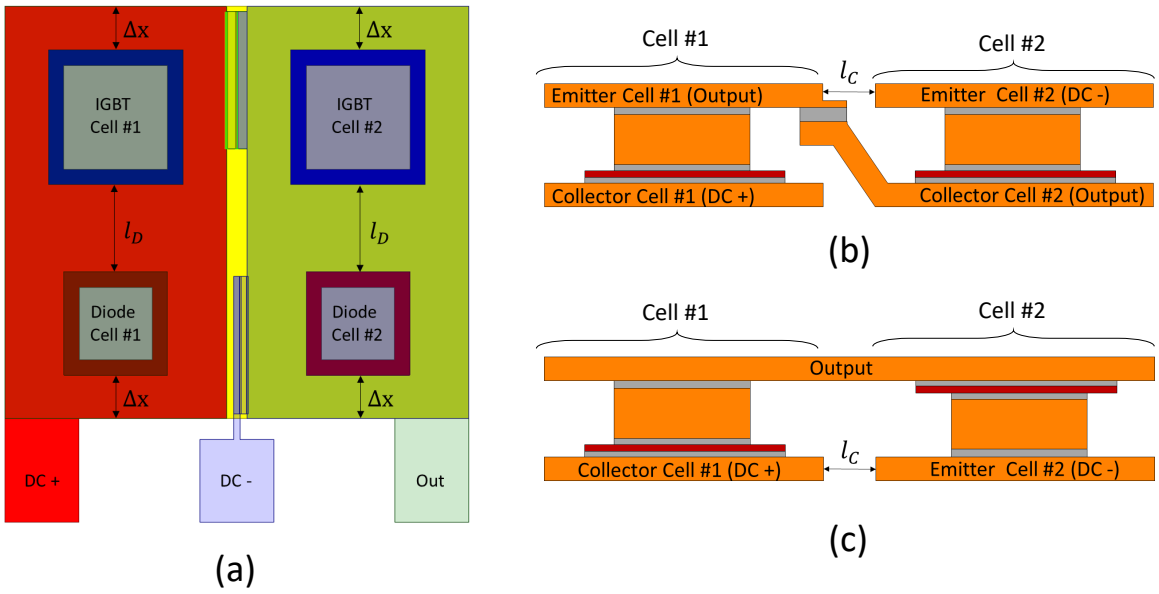


Figure 7.9: Half-bridge inverter packaging architecture choices for DENSO power module. (a) Top-view of two-cells. (b) N-Structure. (b) U-Structure.

sign are cast as 5 categorical variables, namely cooling type (3 choices), bond material (3 choices), ceramic material (4 choices), conductor material (2 choices) and package architecture type (2 choices). The bounds of the continuous variables and categorical choices are given in Table 7.3.

7.3.1 Simulation Setup

The automated optimization framework is given is Figure 7.10. The package architecture, cooling solution, materials used for different layers and physical geometry of the package are determined by the mixed-variable BO method. These are first fed into a finite-element based electromagnetic simulation tool, Ansys Q3D, to extract parasitic power loop inductance, output capacitance and ohmic losses. A thermal simulation is then performed using Ansys Icepak, where the effect of Joule heating from the electrical simulation act as a thermal source along with a 100W loading that is applied to diode and Si insulated-gate bipolar transistor (IGBT) device. The maximum of the junction temperatures obtained from the thermal simulation, along with parasitic L, C and volume of the overall package is then fed back into the optimization algorithm to proceed into the next iteration.

Table 7.3: Mixed-variable sample space considered for power-module optimization.

Parameter		Type	Unit	Min	Max
Plate Thickness	t_{plate}	Cont.	μm	30	70
TIM Thickness	t_{TIM}	Cont.	μm	31	140
Ceramic Thickness	t_C	Cont.	μm	50	500
Metallization	t_M	Cont.	μm	20	35
Spacer Thickness	t_{sp}	Cont.	μm	10	20
Di/IGBT Spacing	l_d	Cont.	μm	300	1200
Plate Clearance	Δx	Cont.	μm	180	960
Cell Seperation	l_C	Cont.	μm	70	770
Package Architecture		Categorical			{N-Structure, U-Structure}
Cooling Solution		Categorical			{Air, Hybrid, Liquid}
Ceramic Material		Categorical			$\{Al_2O_3, AlN, BeO, Si_3N_4\}$
Bond Material		Categorical			{Solder, Ag Sinter, Epoxy Paste}
Conductor Material		Categorical			{Copper, Aluminum}

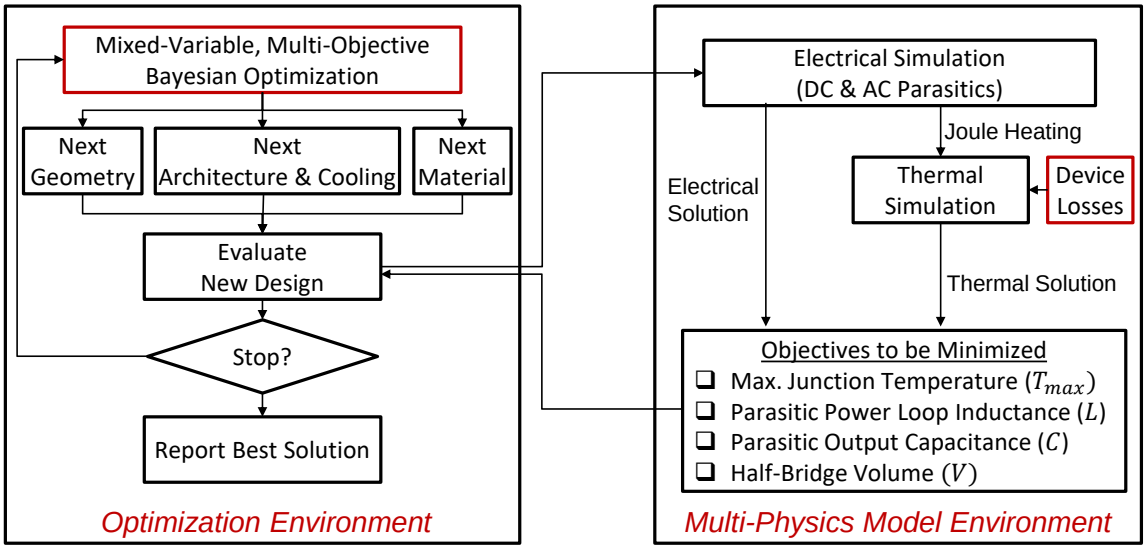


Figure 7.10: Automated optimization setup used for power module optimization.

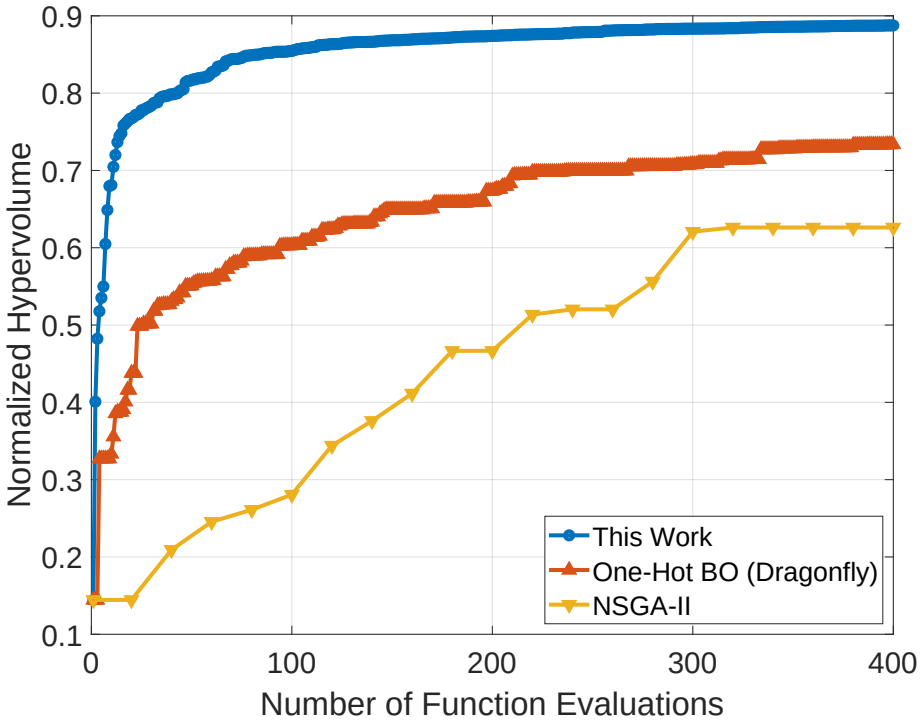


Figure 7.11: Convergence performance of the proposed multi-objective method on power module optimization problem.

7.3.2 Results

Similar to previous section, we compare the performance of the proposed method with NSGA-II and Dragonfly using one-hot encoding to handle categorical variables. We use 20 points determined by LHS as the initial samples and set the optimization budget to 400 function evaluations which corresponds to approximately $\sim$1 day of CPU time ($\sim$4 minutes per query). Figure 7.11 summarizes the hypervolumes of the final Pareto fronts identified by each algorithm along with their convergence performances. The Pareto set identified by the proposed method had a normalized hypervolume of 0.88 as compared to 0.62 and 0.73 for NSGA-II and Dragonfly, respectively, showing a significantly better performance. In addition, the proposed method converged 1.94X and 1.32X faster as compared to NSGA-II and Dragonfly based on the normalized AUC metric. This trend of rapid convergence can also be observed from the sample allocation histogram given in Figure 7.12, which shows that the proposed method focuses its sampling on the most promising categorical choices.

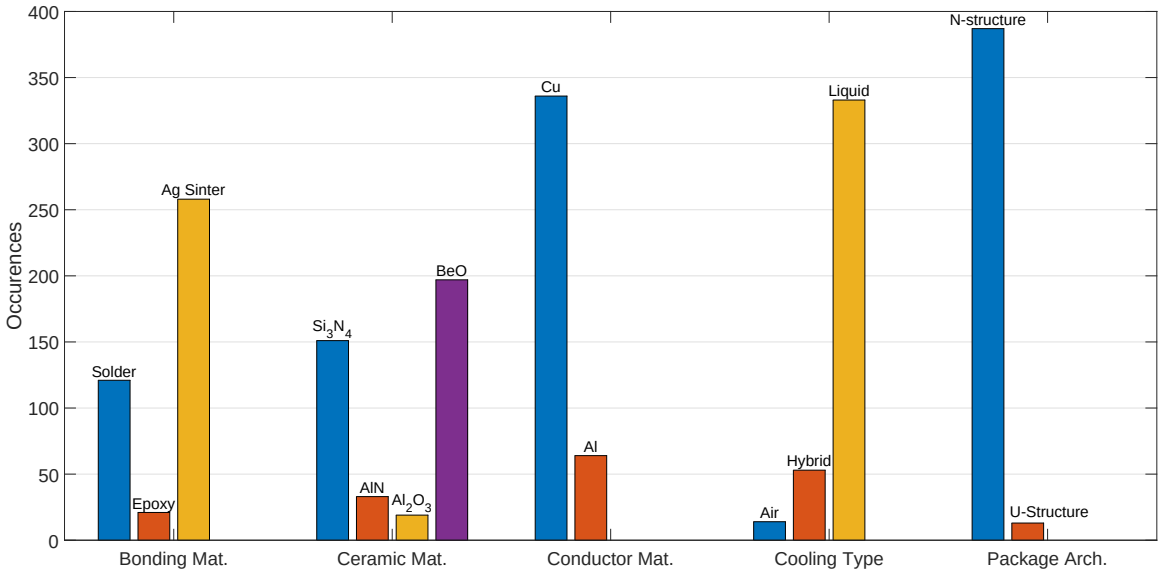


Figure 7.12: Sample allocation of categorical choices done by the proposed algorithm for the fixed budget of 400 function queries.

To analyze the 4-D Pareto set estimated using the proposed method, we use Radial Visualization (RadVis) [108] and project the 4-D objective space into a 2-D radial space. RadVis represents each objective by an anchor. Here, if a point gets farther away from a particular anchor, the value of that corresponding objective becomes lower and a point that prioritizes each objective equally will be in the center of the anchors. The RadVis plot for the four objectives considered in the power module problem is given in Figure 7.13. It can be seen that the estimated Pareto set has a wide diversity in the sense that it contains designs that prioritize electrical or thermal performance over the package volume or vice versa based on the design specifications.

Although RadVis allows to visualize all the designs in the Pareto set, as it is in the projected space, it does not give the spread of each objective that is required to understand the trade-offs involved in the problem. This trade-off analysis can be obtained by plotting different pairs objectives in a regular scatter plot as in Figure 7.14. Here, it can be seen that although there seems to be a trade-off between volume and inductance, the spread of the inductance values in the non-dominated set is only 2.5 nH, which highlights a design opportunity to minimize the volume with the cost of a slight increase in inductance. On

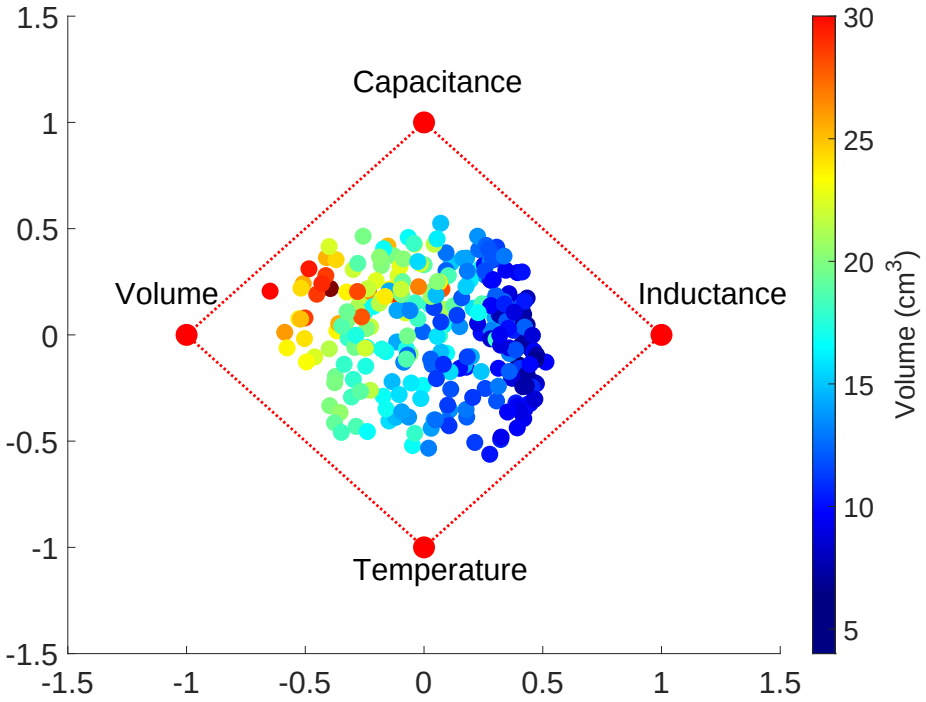


Figure 7.13: Radial visualization (RadVis) of the 4-D Pareto front estimated by the proposed method. The colorbar highlight the package volume of each solution.

the other hand, it can be seen that there is a clear trade-off between volume-temperature, volume-capacitance and temperature-capacitance.

Based on these trade-offs, we pick three different designs from the Pareto set in Figure 7.13 that prioritizes different objectives and compare them to the power module that is designed by hand-tuning [109]. In design #1, we prioritize volume-temperature, whereas in design #2, we prioritize volume-capacitance. In design #3, we prioritized volume and balanced electrical-thermal performance. Performance of these solutions are given in Table 7.4. The design that prioritized volume-capacitance, design #1, has 68.7% lower capacitance and 13.4% reduced package volume as compared to hand-tuned design. On the other hand, design #2 has 15.8% reduced volume and 2.7% less maximum junction temperature than the hand-tuned design while having a better electrical performance. The design that prioritized volume, design #3, has 29.4% reduced volume while having 32.4% less parasitic capacitance. The results show that the Pareto front identified by the proposed method contains a diverse set of designs and can be utilized to make design choices that prioritize

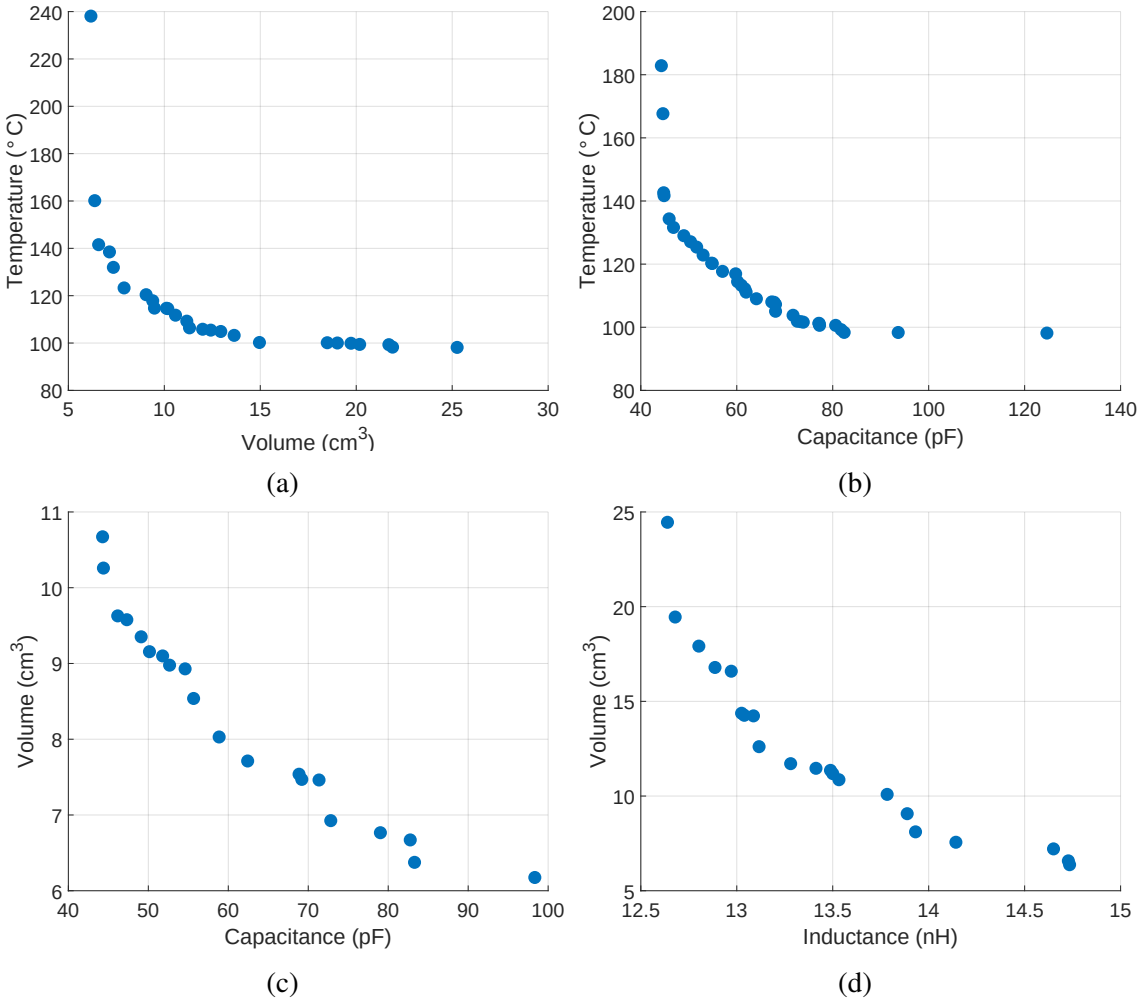


Figure 7.14: Pairwise non-dominated sets of different objectives to facilitate trade-off analysis in power module optimization problem.

Table 7.4: Performance comparison of optimized power modules that prioritize different objectives.

	Hand-Tuned Design [109]	**Design #1**	**Design #2**	**Design #3**
Inductance (nH)	14.9	15.0	14.1	14.7
Capacitance (pF)	163.4	131.3	51.1	110.3
Package Volume (cm^3)	13.44	11.31	11.63	9.48
Max. Temperature (°C)	109.4	106.4	131.5	114.8

different objectives.

7.4 Conclusion

In this chapter, we have proposed a mixed-variable GP model that uses embedding matrices to perform mixed-variable, single- and multi-objective BO of computationally expensive to evaluate blackbox functions. We showed that the categorical domain can be non-linearly transformed into a continuous and bounded latent domain where the conventional GP kernels can be directly used. By embedding this transformation into the kernel function and treating the elements of the embedding matrices as additional kernel hyperparameters, we have showed that the GP can be end-to-end trained to find the latent space that best describes the categorical variables based on their effect on the objective function.

For both single- and multi-objective settings, the proposed method has been applied to a set of synthetic mixed-variable functions with a varying number of continuous and categorical parameters and showed significantly better performance, where a 1.10X-3.74X faster convergence is observed as compared to existing methods in the literature. In addition, we have applied the proposed method to multi-objective and multi-physics optimization of a power electronics module that is used in electrified vehicles, where the sample space consisted of 8 continuous and 5 categorical variables. The designs within the Pareto set found by the proposed method have shown to have a wide diversity which allowed to prioritize

different design objectives. We have showed the optimized designs have up to 2.7% reduced maximum junction temperature, 29.4% reduced package volume and 68.7% reduced parasitic capacitance as compared to a hand-tuned design. The results shown in this chapter shows the applicability of the proposed method to a variety of mixed-variable single- and multi-objective blackbox optimization problems that emerge in science and engineering.

CHAPTER 8

BAYESIAN ACTIVE LEARNING FOR UNCERTAINTY QUANTIFICATION OF ELECTRONIC SYSTEMS

The methodologies described so far in the thesis are geared towards improving the design stage for microelectronic systems through efficient design optimization and design space exploration. Another area that can benefit from Bayesian Learning techniques is *uncertainty quantification* (UQ) of electronic systems to quantify the effect of manufacturing process variations into the design performance. In fact, UQ is becoming increasingly important for high-performance microelectronics as the transistor sizes become less than 10 nm, interposer level line pitches are going into sub-micron region and package level build-up layers are becoming thinner. These pose a variety of challenges to development of reliable manufacturing processes, and often times, the fabricated structure vary from the design parameters. As data rates keeps increasing, these variations can significantly degrade the design performance. Hence, UQ is an essential step for any high-performance microelectronic system.

The easiest way to perform UQ for packaging related applications is to perform Monte Carlo (MC) analysis. Here, a large of number of samples are first collected from the distribution of the uncertain parameters. These samples are then evaluated through the model of the underlying system using SPICE and/or EM solvers to get the probability distribution (PDF) of the electrical performance. MC analysis, however, quickly becomes computationally intractable for modern systems where model evaluations are CPU intensive and a large number of uncertain parameters need to be considered.

A popular way to address these issues is to use polynomial chaos expansion (PCE) [110], where a surrogate model is created to map uncertain input parameters to the output of interest. Then, the MC framework is run on cheap to evaluate PCE model to get the PDF of

the output. However, PCE based approaches suffer from being a deterministic technique. In other words, neither the sensitivity analysis or the output distribution obtained through PCE provide a confidence bound to assess the reliability of the predictions. Further, as the basis functions chosen for PCE relies on the distribution of the input data, one needs to create a new training set every time a new distribution needs to be analyzed. This adds a significant computational overhead when analyzing the effect of different input distributions on the variability of the output.

In this chapter, we present a Bayesian treatment to the UQ of electronic systems. Our objective is to use minimum amount of training data to obtain the following: 1) Output the distribution of interest for the design under evaluation when uncertain parameters have an arbitrarily correlated input distribution, 2) sensitivity analysis to identify which parameters create more variation on the output to provide feedback to process development and 3) the worst-case scenario to ensure the design will be compliant even under all the process variations. Getting the worst-case scenario in particular is the priority objective in UQ for packaging design, such as in signal and power integrity (SI & PI) problems, as systems are commonly designed to perform only slightly better than a given performance margin to reduce the overall cost. Hence, the variations in process, voltage, temperature and material characteristics can easily push the design performance below the performance margin and result in costly design re-spins.

In particular, we introduce a technique called Bayesian Active Learning using Dropout (BALDO) that uses GPs to achieve these objectives along with obtaining confidence bounds associated with the predictions of interest. We first present an alternative training methodology for GPs to make it more appropriate for UQ scenarios, followed by how to exploit this training to get a cost-free sensitivity analysis with confidence bounds, and then provide the sampling strategy used in BALDO to prioritize obtaining worst-case scenario above other objectives.

8.1 Bayesian Training of Gaussian Processes

In the previous chapters of thesis where we used GPs for BO, we have presented that training of a GP corresponds to finding the hyperparameters of the model, θ, through maximizing the log-likelihood as in Equation 2.10. When using an uninformative prior on kernel parameters, this approach is called maximum likelihood estimation (MLE) of the hyperparameters and is the most common method to train GPs. However, observe that the GP posterior obtained by MLE is still conditioned on θ in Equation 2.12. This means that the confidence bounds obtained from the GP assumes a *fixed* hyperparameter vector and only accounts for *data-related uncertainty* but not *parameter-related uncertainty*. The data-related uncertainty in this context refers to both lack of data in specific regions of the sample space and the possible noise associated with observations. This approach works well in practice for BO purposes where the GP is only used as the guide of the optimization.

For UQ purposes, we need a more comprehensive GP model that makes less assumptions since the model needs to be used comprehensively to obtain the output PDF and the corresponding confidence bounds. Hence, the confidence bounds obtained also needs to account for parameter-related uncertainties. This means that instead of assuming a fixed θ, we need to consider all possible θ values, and use a weighted sum of confidence intervals where the bounds obtained with more likely θ values should affect the final confidence bound more than others. Considering weighted sum of all the possibilities effectively means that our predictive confidence bounds no longer depend on θ, hence, is much more robust. This treatment for GPs is called as fully Bayesian approach. The weighted-sum of hyperparameters corresponds to integration in continuous domain and can be written as

$$p(y^* \mid x^*, X_T, Y_T) = \int p(y^* \mid x^*, X_T, Y_T, \theta) p(\theta \mid X_T, Y_T) d\theta \tag{8.1}$$

where $p(y^* \mid x^*, X_T, Y_T)$ is the predictive posterior at a test point x^* that no longer depends on θ and $p(\theta \mid X_T, Y_T)$ is the hyperparameter posterior that acts as the weights. The

challenge with this formulation is analytical intractability of this integration since θ is non-linearly involved in $p(y^* \mid x^*, X_T, Y_T, \theta)$ and $p(\theta \mid X_T, Y_T)$ is not available at all. Here, we resort to Markov Chain Monte Carlo (MCMC) techniques to approximate the integral. MCMC, in particular, slice sampling [111, 112], is an efficient technique to sample from the unavailable weighting function, i.e. the hyperparameter posterior, which is then used to perform MC integration. To sample from an unavailable distribution, we use Bayesian formulation and sample from an equivalent distribution. In our case of hyperparameter posterior, the equivalent distribution can be written as

$$p(\theta \mid X_T, Y_T) \propto p(Y_T \mid X_T, \theta) p(\theta) \tag{8.2}$$

where $p(Y_T \mid X_T, \theta)$ is the GP likelihood in Equation 2.9 and $p(\theta)$ is the prior on θ, i.e. hyperprior. The hyperprior is defined based on domain-knowledge, or can be taken as a uniform distribution in cases where no prior knowledge is available. Sampling from the hyperparameter posterior can also be interpreted as the training of the GP model through learning the hyperparameter posterior. Once the learning is finished, the posterior mean and variance of the GP can be calculated using a mixture of Gaussians formulation. For the posterior mean, this can be written as

$$\begin{aligned} \widehat{\mu}(x^*) =& \mathbb{E}\left[p(y^* \mid x^*, X_T, Y_T)\right] \\ =& \mathbb{E}\left[\mathbb{E}\left[\sum_{n=1}^{N} p(y^* \mid x^*, X_T, Y_T, \theta_n)\right]\right] \\ =& \frac{1}{N}\sum_{n=1}^{N} \widehat{\mu}_{\hat{\theta}_n}(x^*) \end{aligned} \tag{8.3}$$

and for the posterior variance,

$$\begin{aligned}\widehat{\sigma}^2(x^*) =& Var\left[p(y^* \mid x^*, X_T, Y_T)\right] \\ =& Var\left[\mathbb{E}\left[\sum_{n=1}^{N} p(y^* \mid x^*, X_T, Y_T, \theta_n)\right]\right] + \mathbb{E}\left[Var\left[\sum_{n=1}^{N} p(y^* \mid x^*, X_T, Y_T, \theta_n)\right]\right] \\ =& \sigma_e^2 + \frac{1}{N}\sum_{n=1}^{N} \widehat{\sigma}_{\hat{\theta}_n}^2(x^*)\end{aligned} \tag{8.4}$$

where $\hat{\theta}_n$ are the samples from the hyperparameter posterior, i.e., $\theta_n \sim p(\theta \mid X_T, Y_T)$, N is the total number of samples, $\widehat{\mu}_{\hat{\theta}_n}(\cdot)$ and $\widehat{\sigma}^2_{\hat{\theta}_n}(\cdot)$ are the posterior mean and variance for a given θ_n as in Equation 2.13 and Equation 2.14 and σ_e^2 is the empirical variance of sampled posterior means, i.e., $Var\left[\widehat{\mu}_{\hat{\theta}_{1:N}}\right]$.

8.2 Sensitivity Analysis using Hyperparameter Posterior

The learned hyperparameter posterior can be further exploited to obtain a sensitivity analysis with confidence intervals without any further computations, i.e. for free. As indicated in previous chapters, a kernel function with a separate lengthscale parameter for each input parameter as in Table 2.1 allows to the determine the relevance of each parameter to the overall predictions. This is called automatic relevance determination (ARD) kernels. For convenience, we repeat the ARD Matern 5/2 function given in Equation 2.8 here:

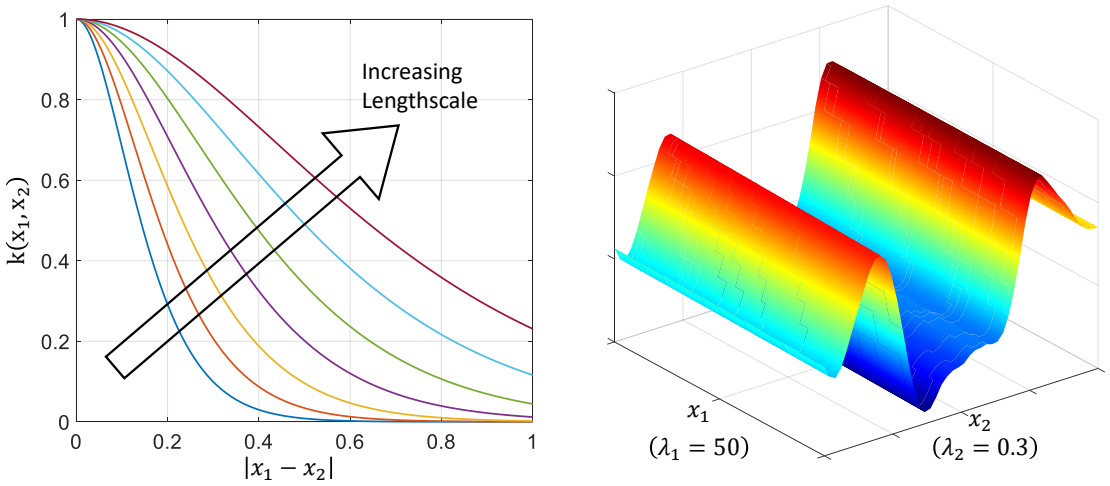
$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2(1 + \sqrt{5}r + \frac{5}{3}r^2)e^{-\sqrt{5}r} \tag{8.5}$$

where

$$r = \left(\sum_{d=1}^{D} \frac{(x_{i,d} - x_{j,d})^2}{\lambda_d^2}\right)^{1/2}$$

and $x_{.,d}$ is the d^{th} dimension of input parameter vector, λ_d is the lengthscale of each input parameter and σ_f is the scaling constant.

The effect of the lengthscale parameter on defining the correlation between inputs of



(a) (b)

Figure 8.1: Illustration of lengthscale based sensitivity analysis. (a) Effect of increasing lengthscale on the covariance function, showing a longer lengthscale corresponds to shorter distance between x_1 and x_2. (b) A 2D prior from GP where the parameter with shorter lengthscale has significantly higher impact than the other.

two data samples, $\mathbf{x}_1$ and $\mathbf{x}_2$, is illustrated here in Figure 8.1. The covariance function in GPs quantifies the relevance between two locations in the sample space. If the locations are closer to each other, then it can be said they are highly correlated. The lengthscale parameter is the scaling factor of this distance between two points. In other words, it determines at what distance such two such points in the sample space become uncorrelated to each other. A short lengthscale then means that points in close proximity are uncorrelated, indicating that even a small change of that parameter causes a big impact in the output. A separate lengthscale per input parameter implements this in a directional manner. As can be seen in Figure 8.1b for the 2D case, when the lengthscale of the first parameter, λ_1, is significantly shorter than of the second parameter, λ_2, the function varies more in the direction of the first parameter.

If we take the inverse of this lengthscale and normalize across all parameters such that their sum equals to 1, we obtain the percent impact, i.e. sensitivity, of each input parameter. Observe that in the MCMC framework, the learned hyperparameter posterior includes the joint posterior over the lengthscales, i.e. $p(\lambda_1, ..., \lambda_D \mid X_T, Y_T)$. If we marginalize this joint

distribution as $\{p(\lambda_1 \mid X_T, Y_T), ..., p(\lambda_D \mid X_T, Y_T)\}$, we get the posterior distribution over first-order impact of each parameter. Similarly, the full joint posterior provides D^{th}-order impact. The inverse mean of the hyperparameter posterior then can becomes *predicted sensitivity* of each parameter for an arbitrary order, whereas the standard deviation of the posterior becomes the *uncertainty over impacts*. Such formulation is unique to Bayesian treatment to sensitivity analysis and provides invaluable information as we later show in this chapter.

8.3 Bayesian Active Learning using Dropout

Active learning is a general name given to adaptive techniques that exploits the information obtained from a model to determine the next set of parameters to be simulated to achieve a certain objective. When the goal is global optimization using a probabilistic model, it becomes the BO framework. When the goal is to build a predictive probabilistic model (such as a GP) with minimum amount of data, it is called Bayesian Active Learning (BAL). In BAL, the next sampling point is selected to decrease the prediction uncertainty of the model. As a result of this adaptive sampling strategy, the final model can achieve a better predictive accuracy as compared to apriori design of experiments (DoE) based methods where training data is collected without the knowledge of the system response.

A common BAL strategy is to select next sampling point to have highest entropy. For GPs, this can be written in closed form as:

$$H(x^*) = 0.5 \log \left(2\pi e \sigma^2(x^*)\right) \tag{8.6}$$

where $\sigma^2(x^*)$ is the posterior variance of GP. Another popular strategy is the mutual-information (MI) criteria [113], but calculating MI is a computationally intensive approach for high-dimensional problems. Hence, we focus on easy to calculate entropy criterion.

As the entropy criterion in Equation 8.6 aims to minimize the uncertainty of the model

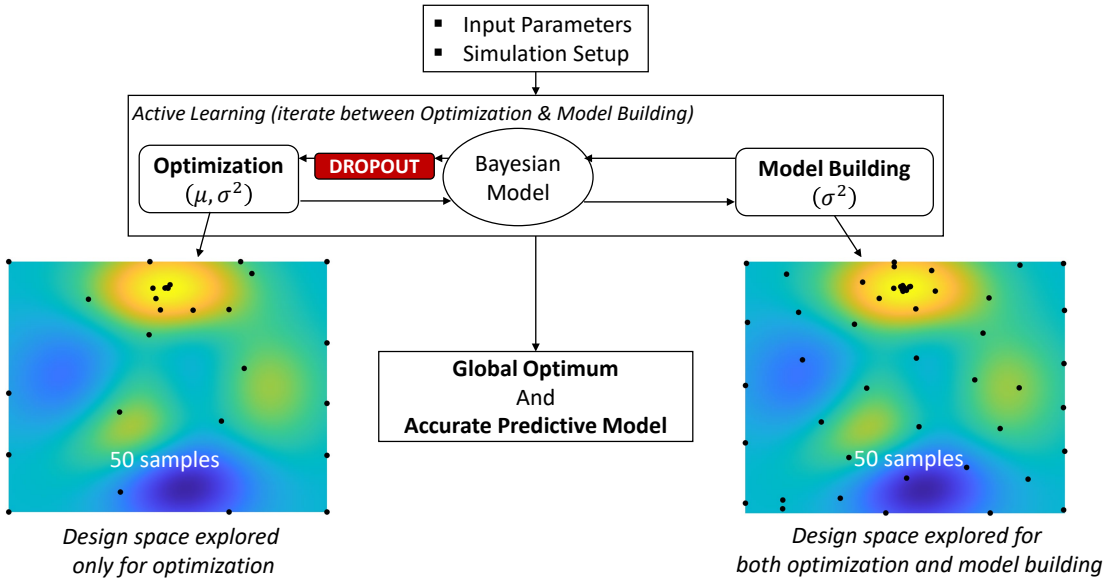


Figure 8.2: Simultaneous optimization and model building using BALDO.

rather than regions that are likely to contain the global optima, it is prone to missing the actual worst-case scenario. As the worst-case scenario is of utmost importance in UQ for packaging problems, a seperate BO framework needs to be adapted to ensure finding the worst-case. On the other hand, if a direct BO is performed, the final predictive model obtained will only be accurate in regions that are likely to contain the worst-case scenario rather than the whole sample-space as required to get PDF of the output. Hence, a complete UQ for packaging problems requires a BAL followed by BO or vice versa to achieve all the objectives.

Instead of performing a BAL followed by BO or vice versa for a complete UQ, we introduce a new technique called Bayesian Active Learning using Dropout (BALDO). Here, we introduce the concept of simultaneous model building and optimization as in Figure 8.2. The goal in BALDO is to jointly learn the underlying system to derive an accurate predictive model over whole sample space and converge to the worst case scenario to ensure design compliance. To achieve this, we approach the active learning problem in two stages, namely *optimization stage* and *learning stage* and we sequentially alternate between these stages at every iteration. The underlying idea here is that although global optimization and

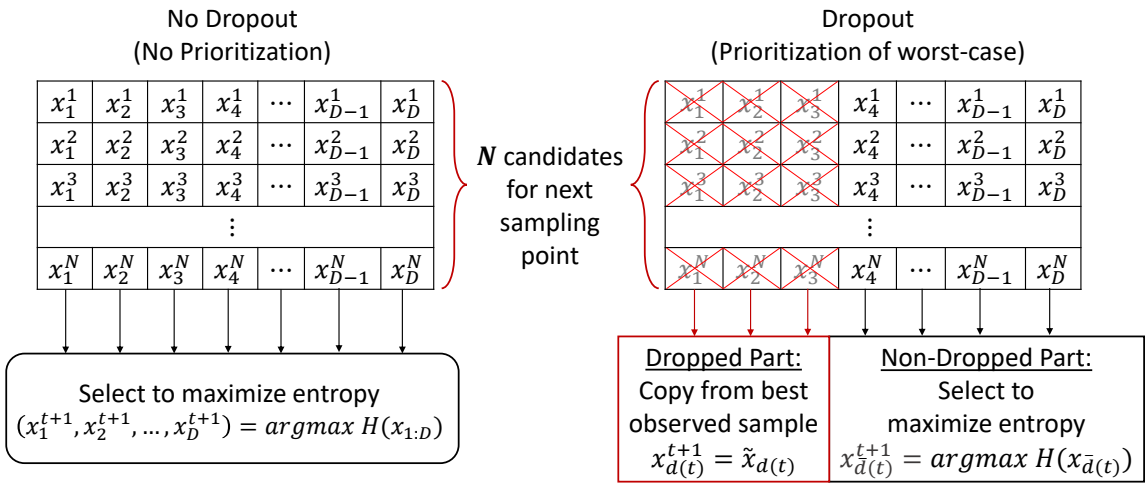


Figure 8.3: The dropout over parameters strategy used in BALDO to prioritize finding the worst-case scenario.

deriving an accurate predictive model are two different goals, they complement each other. If the GP model that guides the BO has better predictive accuracy, then the convergence rate significantly increases. For predictive modeling, obtaining the extrema points allows to learn better hyperparameters of the GP.

In the optimization stage, we follow a BO approach and select the next sampling point that is likely to be the global minima by using the learning acquisitions functions strategy of TSBO. After the next point is determined and simulation is performed, the GP model is trained using MCMC approach in Equation 8.1 and the current best input vector, $\widetilde{x}_D = \{x_1, x_2, ..., x_D\}$, is passed to the learning stage.

In the learning stage, the objective is to explore the sample space such that the uncertainties about regions other than the ones focused by optimization stage are also reduced. This can be achieved by selecting the point maximizing entropy in Equation 8.6. However, as we aim to prioritize finding the worst case scenario, we introduce dropout to determine the next sampling point, x^{t+1}. Here, we first randomly select a group of d dimensions out of the total D dimensions of the current best point, $\widetilde{x}_D$, and denote this group as $d(t)$. Then, we copy the parameters in $d(t)$ to the new sampling point, i.e. $x_{d(t)}^{t+1} = \widetilde{x}_{d(t)}$. The remaining dimensions, $\bar{d}(t) = (D \setminus d(t))$, are selected such that they maximize the entropy in Equa-

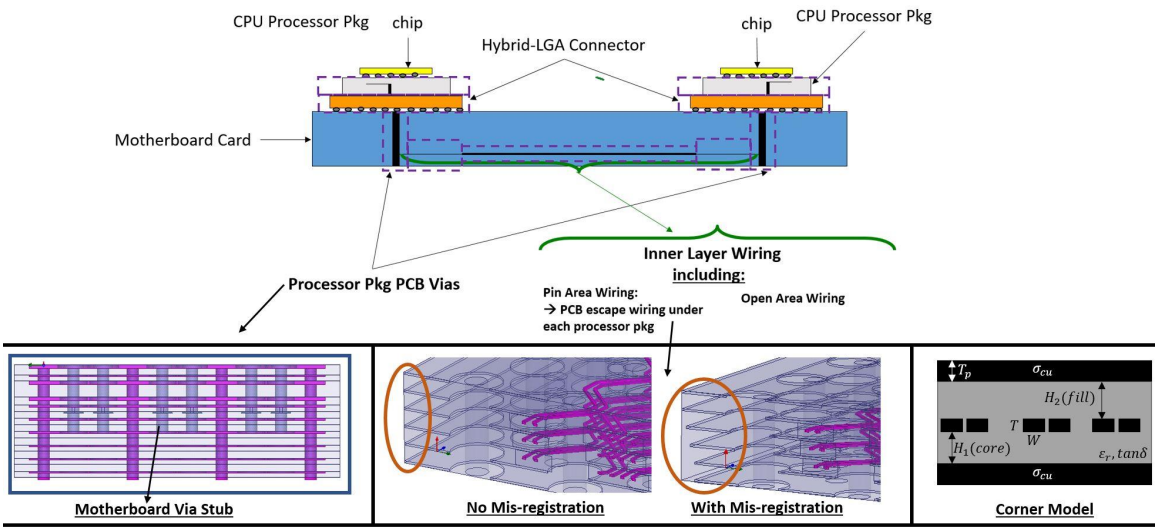


Figure 8.4: Structure of the IBM POWER9 processor to processor X-Bus channel to illustrates uncertain parameters under consideration.

tion 8.6. The simulation is then performed at $x_{t+1} = x^{t+1}_{\bar{d}(t)} \cup x^{t+1}_{d(t)}$, the GP is re-trained using the Bayesian formulation as in Section 6.1 and we proceed to the next iteration using the optimization stage. This prioritization of obtaining the worst-case scenario through dropout is illustrated in Figure 8.3.

8.4 Application to Uncertainty Quantification of High-Speed Channel Signaling

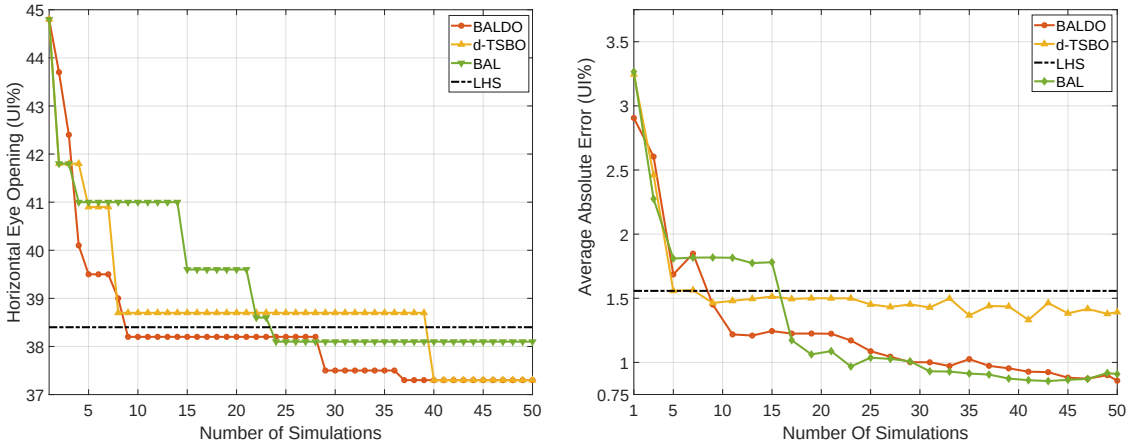
We now apply BALDO to perform a complete UQ of an high-speed channel to analyze its signaling performance under various process variations. We consider a comprehensive industrial example, IBM's POWER9 processor to processor X-Bus channel [114]. The structure of the channel is given in Figure 8.4 and it operates at a data rate of 16 Gb/s using differential signaling. The UQ problem is posed as a 6 dimensional problem, where the uncertain variables include motherboard impedance and attenuation corners, maximum backdrilled via stub length, amount of pin area wiring mis-registration and TX & RX CPU package corner impedance. We consider a discrete sample space as in Table 8.1 to take ad-

Table 8.1: Uncertain Parameters of the High-Speed Channel

Parameters	Values
Motherboard Impedance Corner	Low, Nominal, High
Motherboard Attenuation Corner	Low, Nominal, High
Backdrilled Via Stub Length (mm)	0.1016, 0.2032, 0.3048, 0.4064, 0.5082
Pin Area Wiring Mis-registration (mils)	0, 1.25, 2.5, 3.75, 5.00
TX CPU Package Corner Impedance	Low, Nominal, High
RX CPU package Corner Impedance	Low, Nominal, High

vantage of pre-computed S-Parameters and corner cases. The objective here is to find the channel variable combination with the minimum horizontal eye opening (HEYE) while accurately estimating the PDF of HEYE and perform sensitivity analysis to determine which variables have higher effect on its variability.

The simulation framework used consists of three main parts, namely generating frequency response, optimizing equalization setting and finally generating the eye diagram. First, 36-port S-Parameters of individual components comprising end-to-end channel are cascaded to each other. These components include μ-bumps, TX & RX CPU package, hybrid land grid array (LGA) connector, motherboard via array, pin area and open-area wiring. Each component itself is represented by cascade of multiple S-Parameters, which are simulated using full-wave EM solvers prior to performing UQ. The full channel S-Parameter includes 1 victim differential pair and 8 crosstalk aggressors. The overall S-Parameter of the channel is then exported to IBM's high speed serial clock data recovery (HSSCDR) simulation tool, an in-house time domain link simulator, to be combined with behavioral models of the driver/receiver circuitry. At the TX side, an adaptive feed forward equalization (FFE) is used and at the RX side, a 12 tap decision feedback equalizer (DFE), an amplifier with automatic gain control (AGC) and a continuous-time linear equalizer (CTLE) are used. The settings for AGC and CTLE are optimized for each channel combination by sweeping all possible combinations in time domain. Finally, these equalization settings are used in a 10 million bit time-domain simulation to characterize the horizontal eye opening



(a) (b)

Figure 8.5: Convergence comparison of different methods for UQ of high-speed signaling. (a) Predictive Accuracy. (b) Worst-case HEYE.

Table 8.2: Comparison of Different Methods for UQ

	LHS	**d-TSBO**	**BAL**	**BALDO**
Av. Abs. Error (UI%)	1.55	1.39	0.909	0.857
Min. HEYE (UI%)	38.4	37.3	38.1	37.3

at a BER of 10^{-12}. Although the sample space covers only 1125 different channel combinations, each simulation takes approximately 45 minutes when parallelization over 16 CPU cores are applied where applicable. Total simulation time then aggregates to 35 CPU days and even a better parallelization scheme with more computational resources can only reduce this time to approximately 10 CPU days.

The performance of BALDO is compared to performing only optimization using discrete version of the TSBO algorithm (d-TSBO), a BAL algorithm where the next sample is chosen to have highest entropy in Equation 8.6 and Latin Hypercube Sampling (LHS), where a GP model is trained with 50 samples determined apriori as conventional DoE based surrogate model development. The predictive accuracy of each model is obtained through validating them over 110 randomly collected samples.

The results are summarized in Table 8.2 and convergence curves are given in Figure 8.5. After 50 simulations in approximately 1.5 CPU days (down from 35 CPU days with equiv-

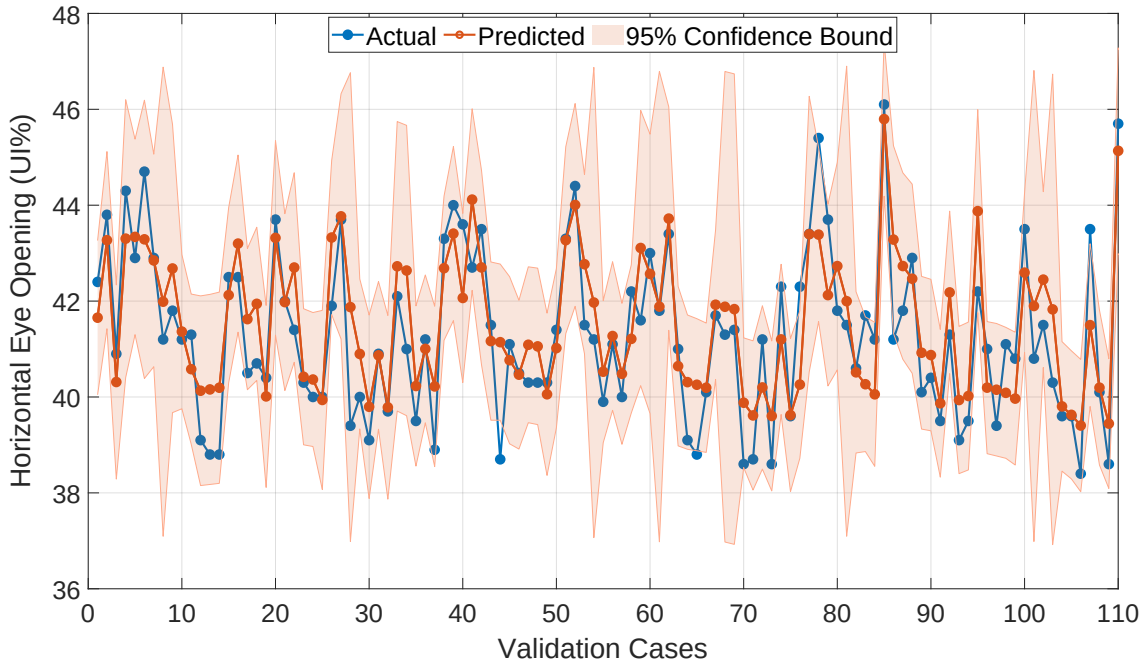


Figure 8.6: Predicted HEYE and confidence intervals for 110 validation cases using the GP obtained with BALDO.

alent resources), BALDO provided a minimum HEYE of 37.3% unit interval (UI) with 0.86%UI average absolute prediction error on the validation set. Although BAL provided a similar but slightly worse prediction error of 0.91%UI, it could not identify the worst scenario, which was the main objective as depicted in previous sections. Performing only optimization using d-TSBO converged to same worst-case scenario as BALDO, but as expected, with a poor predictive performance with an average error of 1.39%UI. On the other hand, the model derived using LHS, the only non-active learning method, performed worst compared to other techniques in predictive performance with 1.55%UI error and could not identify the worst case channel.

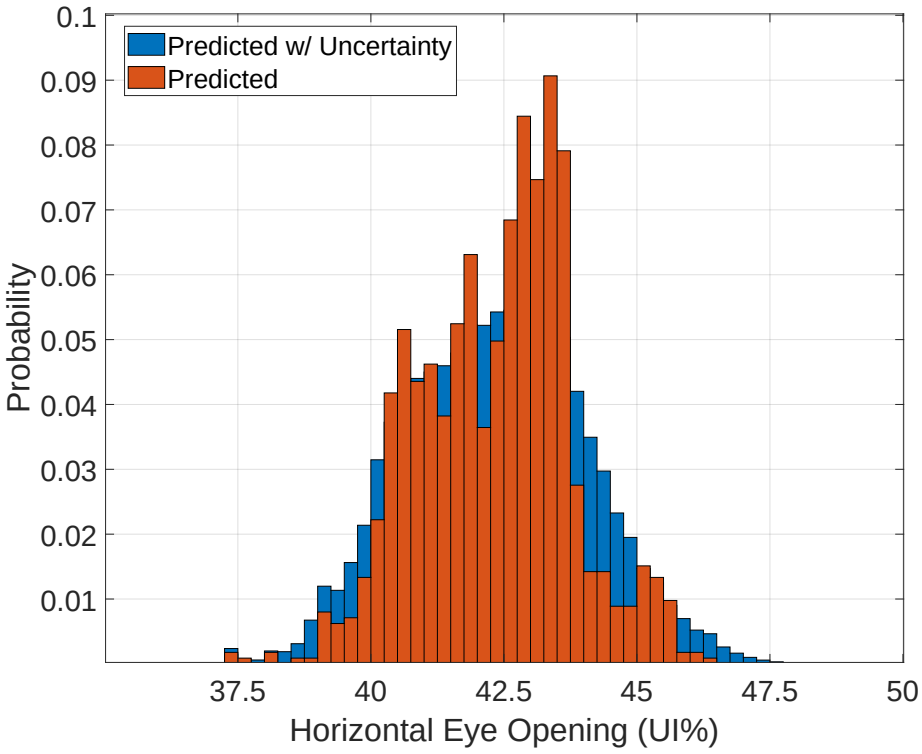
The predictive performance of the final GP model obtained through BALDO is also provided in Figure 8.6 along with corresponding confidence intervals for all validation cases. When we use the GP obtained through BALDO to predict all other channel combinations, we find that average 95% confidence interval around the predictions to be ± 3.1%UI. More importantly, we observe that all the test cases are within the 95% confidence intervals

around the predictions, showing the quality and reliability of the model. In terms of run times, it took $\sim$37.6 hours to collect the 50 simulations and $<$1 minute to train the GP model. Once trained, the GP model can predict HEYE of all 1125 channel combinations and the confidence bounds in $\sim$0.7 seconds as compared to 35 days using the aforementioned simulation framework. If the model uncertainty is greater than an acceptable margin, more simulations can be performed to reduce the width of the confidence intervals, thereby answering the question on how many data samples are required to derive a reliable model.

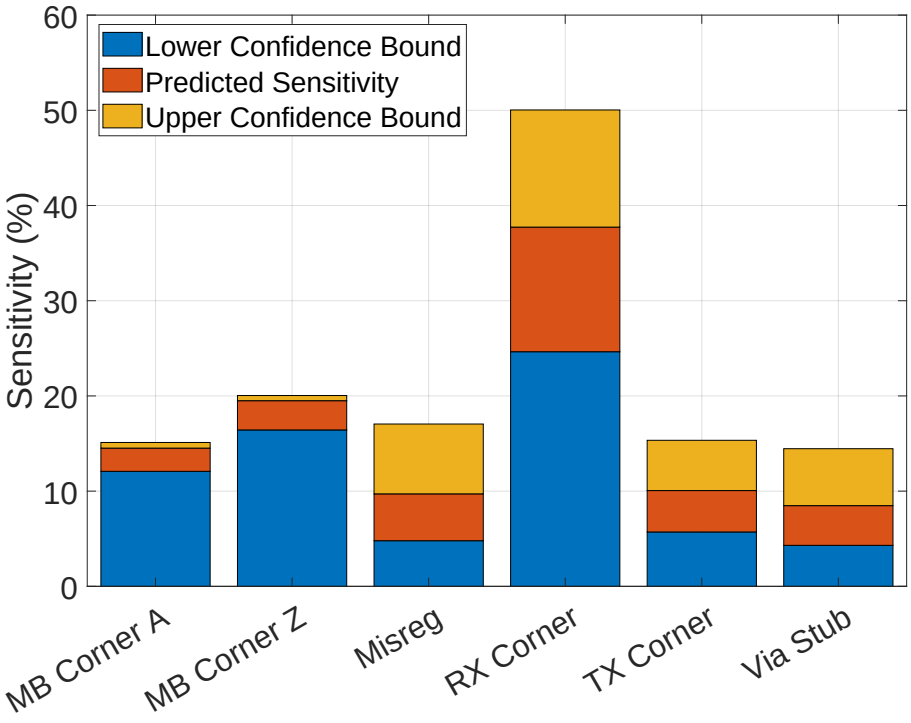
In addition, Figure 8.7 shows the PDF of HEYE and sensitivity analysis of input variables using the final model derived by BALDO after 50 simulations. As the data is very scarce, the confidence intervals associated with the predictions provide a highly valuable interpretation of the model. As we prioritized finding the worst-case channel, the sampling was more focused on regions that are more likely to contain the worst-case scenario. This can be seen from the PDF given in Figure 8.7a, as it shows the model is more confident with the lower tail of PDF compared to its upper tail, showing only 3.01% probability of finding a worse case scenario. The sensitivity analysis given in Figure 8.7b shows the model confidently selects CPU RX package corner impedance as the dominant variable that affects HEYE since lower confidence bound (LCB) of its weight is higher than upper confidence bound (UCB) of the others. However, high uncertainty associated with the prediction of its weight indicates more data is required to calculate its exact weight.

8.5 Conclusion

In this chapter, we have introduced a new algorithm, Bayesian Active Learning using Dropout (BALDO), that simultaneously creates an accurate predictive probabilistic model and optimizes - finds the worst-case - of a blackbox function to perform a complete uncertainty quantification of electronic systems and demonstrated it on an industrial high-speed channel. By jointly performing learning and optimization using BALDO, we have shown that both optimization performance and predictive accuracy of the model can be increased



(a)



(b)

Figure 8.7: The predicted statistics of the channel along with confidence bounds. (a) Histogram of HEYE. (b) Sensitivity Analysis.

compared to BAL and d-TSBO, which focus on only learning and optimization, respectively. By leveraging Bayesian inference and exploiting hyperparameter posterior, we have shown that the BAL-DO method can generate sensitivity analysis without any extra cost. Although we have applied BALDO to high-speed channels, the results shown in this chapter demonstrates the algorithms applicability to other systems.

CHAPTER 9
SUMMARY AND FUTURE WORK

9.1 Dissertation Summary

In this thesis, we have investigated the application of machine learning (ML) techniques to semiconductor packaging design. We have developed new neural network architectures, along with Bayesian learning based models and algorithms for accurate system-level design optimization, design space exploration (DSE) and uncertainty quantification (UQ) for high-performance packaging and demonstrated their effectiveness on emerging design applications. The contributions of this thesis as presented in each chapter can be listed as:

Chapter 3 presented a new convolutional based neural network (NN) architecture, namely Spectral Tranposed Convolution Network (S-TCNN), to predict high-dimensional frequency responses given their corresponding design parameters. The new architecture exploits the spatial correlation in the frequency axis, i.e. neighboring frequency points being correlated with each other, to address the high output dimensionality in a computationally- and data-efficient manner without losing model representation capability. In addition, we have shown how S-TCNN model can be extended to include confidence intervals around the predictions by making use of approximate variational Bayesian learning techniques, where the resulting model was named as Bayesian S-TCNN. We have demonstrated the performance of S-TCNN model on predicting the frequency response of solenoidal inductors with magnetic cores, and performance of Bayesian S-TCNN on predicting the frequency response of radar cross section (RCS) of an aircraft.

Chapter 4 presented development of physically consistent NNs to parameterize broadband S-Parameters and ensure NN predicted S-Parameters to be physically consistent, i.e. passive and causal. In particular, we have presented two new layers that can be used in

a NN architecture, namely causality enforcement layer (CEL) and passivity enforcement layer (PEL). The CEL uses the Hilbert Transform to relate the real and imaginary parts of the NN predicted S-parameters, while the PEL ensures that the singular values of the predicted S-parameters to be less than 1. Both CEL and PEL can be appended to any existing NN architecture as the last two layers, thereby enabling end-to-end training to learn a physical representation of the underlying data. We have combined CEL and PEL with the S-TCNN architecture and showed its performance on three design applications. Furthermore, we have shown that CEL and PEL can be combined with Bayesian S-TCNN model to learn a physically consistent posterior distribution in the sense that every broadband S-Parameter sample from the learned posterior holds causality and passivity.

Chapter 5 presented a new design optimization technique called Two-Stage Bayesian Optimization (TSBO). The TSBO algorithm is developed specifically to address the challenges in optimization of electronic systems rather than being a general purpose optimization method. Specifically, we have developed a novel hierarchical partitioning tree based sampling strategy to perform efficient optimization in large sample spaces with moderate dimensionality ($D \approx 10$) while eliminating auxiliary optimization from the BO framework. In addition, we have introduced a sub-learning strategy to the overall BO framework to learn which acquisition function performs the best for the given problem to make TSBO applicable to different design optimization problems. We have demonstrated the effectiveness of TSBO on two design applications, namely clock skew minimization 3D ICs and optimization of integrated voltage regulators (IVR) that are equipped with solenoid inductors with magnetic cores, where we have shown that TSBO outperforms state-of-the-art BO and non-BO methods as well as hand-tuned designs.

Chapter 6 presented a new high-dimensional BO method called Bayesian Optimization with Deep Partitioning Tree (DPTBO). Here, we have identified challenges related to scaling BO methods to high-dimensional problems in the field of high-frequency electronics.

The DPTBO method is then developed by making the appropriate statistical assumptions for optimization of high-frequency electronics and developing a sampling strategy around it. In particular, we have used an Additive Gaussian Process (ADD-GP) as the probabilistic model in the BO framework to explicitly capture lower-order interactions in the high-dimensional sample space. Unlike other methods in the literature that assume input parameters can be grouped into low-dimensional disjoint sets that do not interact with other, the kernel we used captured interactions between every input parameter pair. To address algorithmic challenges related to high-dimensional problems, we have developed a Deep Partitioning Tree technique that extends hierarchical partitioning tree based approaches to high-dimensional sample spaces by utilizing the sensitivities of each parameter. We have demonstrated the effectiveness of DPTBO on three design applications, namely eye diagram optimization for high-speed channels, minimizing losses in sub-THz air-filled substrate integrated waveguides and wireless power transfer based power delivery for IoT applications. We have shown that DPTBO outperform existing BO and non-BO methods in terms of convergence and final design performance.

Chapter 7 presented a new mixed-variable (categorical & continuous) GP model that uses embedding matrices to perform mixed-variable, single- and multi-objective BO for multi-physics design optimization. Here, we designed a new GP model where we use embedding matrices followed by a non-linear function to transform categorical parameters into a continuous and bounded latent domain that allows the use of conventional GP kernels. This transformation is performed at the kernel level of the GP, and the elements of the embedding matrices are treated as hyperparameters of the GP model and are jointly learned along with kernel function parameters through GP likelihood. We then present a new sampling strategys that can be combined with the presented mixed-variable GP model for multi-objective BO. We demonstrate the effectiveness of the proposed method on a variety of synthetic test function along with a comprehensive design application, namely multi-physics co-optimization of architecture, material and layout of an half-bridge inverter package as used

in electric vehicles.

Chapter 8 presented a new method to perform uncertainty quantification (UQ) of electronic systems, named as Bayesian Active Learning using Dropout (BALDO). The BALDO method is developed to use minimum amount of training data to perform a complete UQ to obtain: a) Output the distribution of interest for the design under evaluation when uncertain parameters have an arbitrarily correlated input distribution, b) sensitivity analysis to identify which parameters create more variation on the output to provide feedback to process development and c) the worst-case scenario to ensure the design will be compliant even under all the process variations. To achieve all three objectives in a single and automated framework, we have introduced a new active learning based sampling strategy to simultaneously build a model with high-predictive accuracy (for (a) and (c)) and optimize (minimize) the underlying function to find the worst-case scenario. We have demonstrated the performance of the proposed BALDO method for UQ of high-speed signaling in a commercial high-speed channel and shown that it significantly outperforms the existing techniques.

9.2 Publications

The material presented in this thesis has resulted in following publications.

Book Chapter:

1. **H. M. Torun**, M. Larbi and M. Swaminathan, *"Machine Learning based Optimization and Uncertainty Quantification for Integrated Systems"*, in Machine Learning in VLSI CAD, Springer, Editors: Ibrahim Elfadel, Duane Boning & Xin Li, 2019.

Journal Publications:

1. **H. M. Torun**, R. Wong, D. Lohan, S. Joshi, H. Ukegawa, V. Smet, M. Swaminathan, *"A Mixed-Variable Multi-Objective Bayesian Optimization Method and its Application to Power Module Design"*, in IEEE Access, under review.

2. O. Bhatti, **H. M. Torun**, M. Swaminathan, *"Machine Learning Framework for Extrapolation of Frequency Response"*, in IEEE Transactions on Components, Packaging, and Manufacturing Technology (CPMT), under review.

3. J. Kim, G. Murali, H. Park, E. Qin, H. Kwon, V. Chekuri, N. Rahman, N. Dasari, A. Singh, M. Lee, **H. M. Torun**, K. Roy, M. Swaminathan, S. Mukhopadhyay, T. Krishna, S. Lim, *"Architecture, Chip, and Package Codesign Flow for Interposer-Based 2.5-D Chiplet Integration Enabling Heterogeneous IP Reuse"*, in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020.

4. M. Swaminathan, **H. M. Torun**, H. Yu, J. Hejase, D. Becker, *"Demystifying Machine Learning for Signal and Power Integrity Problems in Packaging"*, IEEE Transactions on Components, Packaging, and Manufacturing Technology (CPMT), 2020.

5. **H. M. Torun**, A. Durgun, K. Aygun, M. Swaminathan, *"Causal and Passive Parameterization of S-Parameters using Neural Networks"*, IEEE Transactions on Microwave Theory and Techniques (MTT), 2020.

6. M. Lee, A. Singh, **H. M. Torun**, J. Kim, S. Lim, M. Swaminathan, S. Mukhopadhyay, *"Automated I/O Library Generation for Interposer-based System-in-Package Integration of Multiple Heterogeneous Dies"*, IEEE Transactions on Components, Packaging and Manufacturing Technology (CPMT), 2019.

7. S. Mukhopadhyay, Y. Long, C. S. Nair, B. H. DeProspo, **H. M. Torun**, M. Kathaperumal, V. Smet, B. Mudassar, D. Kim, S. Yalamanchili, M. Swaminathan, *"Heteregenous Integration for Artificial Intelligence: Challenges and Opportunities"*, IBM Journal of Research and Development, 2019. [119]

8. **H. M. Torun**, M. Swaminathan, *"High Dimensional Global Optimization Method for High-Frequency Electronic Design"*, IEEE Transactions on Microwave Theory and Techniques, 2019.

9. C. A. Pardue, **H. M. Torun**, M. L. F. Bellaredj, M. Swaminathan, *"RF Near-Field Coupling System Using Reverse PDN and Considering Electromagnetic Effects"*, IEEE Transactions on Electromagnetic Compatibility, 2019.

10. R. Trinchero, M. Larbi, **H. M. Torun**, F. G. Canavero and M. Swaminathan, *"Machine Learning and Uncertainty Quantification for Surrogate Models of Integrated Devices With a Large Number of Parameters"*, IEEE Access, 2018.

11. C. Pardue, M. Bellaredj, **H. M. Torun**, M. Swaminathan and A. K. Davis, *"RF Wireless Power Transfer using Integrated Inductor"*, IEEE Transactions on Components, Packaging and Manufacturing Technology, 2018.

12. **H. M. Torun**, M. Swaminathan, A. K. Davis, M. L. F. Belladredj, *"A Bayesian based Global Optimization Algorithm and its Application to Integrated Systems"*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2018.

Conference Publications:

1. J. Kim, V. Chekuri, N. Rahman, M. Dolatsara, **H. M. Torun**, M. Swaminathan, S. Mukhopadhyay, S. Lim, *"Silicon vs Organic Interposer: PPA and Reliability Tradeoffs in*

Heterogeneous 2.5D Chiplet Integration", 38th IEEE International Conference on Computer Design (ICCD), October 2020. **(Best Paper Award)**

2. X. Yang, J. Tang, **H. M. Torun**, W. Becker, J. Hejase, M. Swaminathan, *"Rx Equalization for a High-Speed Channel Based on Bayesian Active Learning Using Dropout"*, in IEEE 29th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2020.

3. K. Scharff, **H. M. Torun**, C. Yang, M. Swaminathan, C. Schuster, *Bayesian Optimization for Signal Transmission Including Crosstalk in a Via Array"*, in IEEE International Symposium on Electromagnetic Compatibility (EMC Europe), 2020.

4. H. Yu, **H. M. Torun**, M. Rehman, M. Swaminathan, *"Design of SIW Filters in D-Band using Invertible Neural Nets"*, IEEE International Microwave Symposium (IMS), 2020.

5. **H. M. Torun**, A. C. Durgun, K. Aygun, M. Swaminathan, *"Enforcing Causality and Passivity of Neural Network Models of Broadband S-Parameters"*, IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2019.

6. K. Roy, M. Ahadi, **H. M. Torun**, R. Trinchero, M. Swaminathan, *"Inverse Design of Transmission Lines with Deep Learning"*, IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2019.

7. **H. M. Torun**, H. Yu, N. Dasari, V. C. K Chekuri, A. Singh, J. Kim, S. Lim, S. Mukhopadhyay, M. Swaminathan, *"A Spectral Convolutional Net for Co-Optimization of Embedded Inductors and Integrated Voltage Regulators"*, International Conference on Computer-Aided Design (ICCAD), 2019.

8. J. Kim, G. Murali, H. Park, E. Qin, H. Kwon, V. C. K. Chekuri, N. Dasari, A. Singh, M. Lee, **H. M. Torun**, K. Roy, M. Swaminathan, S. Mukhopadhyay, T. Krishna, S. Lim, *"Architecture, Chip, and Package Co-design Flow for 2.5D Integration of Reusable IP Chiplets"*, ACM Design Automation Conference (DAC), 2019.

9. **H. M. Torun**, N. Dasari, A. Singh, M. Lee, J. Kim, H. Park, H. Kwon, E. Qin, T. Krishna, S. Lim, S. Mukhopadhyay, M. Swaminathan, *"Design Space Exploration of Power Delivery in Heterogeneous Integration"*, Government Microcircuit Application and Critical Technology Conference (GOMACTech), 2019.

10. J. Kim, E. Qin, H. Park, **H. M. Torun**, M. Swaminathan, T. Krishna, S. Lim, *"Enabling Heterogeneous IP Reuse with Interposer-based 2.5D ICs and Custom Interface Protocol"*, Government Microcircuit Application and Critical Technology Conference (GOMACTech), 2019.

11. M. Lee, A. Singh, **H. M. Torun**, J. Kim, S. Lim, M. Swaminathan, S. Mukhopadhyay, *"Automated Generation of All-Digital I/O Library Cells for Multiple Dies in System-in-Package Integration"*, Government Microcircuit Application and Critical Technology Conference (GOMACTech), 2019.

12. M. Lee, J. Kim, A. Singh, **H. M. Torun**, M. Swaminathan, S. Lim, S. Mukhopadhyay, *"On the design of energy-efficient I/O circuits for interposer-based 2.5D system-in-Package"*, IEEE SOI-3D-Subthreshold Microelectronics Technology Unified Conference (S3S), 2018.

13. **H. M. Torun**, J. A. Hejase, J. Tang, W. D. Becker and M. Swaminathan, *"Bayesian Active Learning for Uncertainty Quantification of High Speed Channel Signaling"*, IEEE 27$^{\text{th}}$ Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2018. (**Best Student Paper Award**)

14. M. Lee, A. Singh, **H. M. Torun**, J. Kim, S. Lim, M. Swaminathan, and S. Mukhopadhyay, *"Automated Generation of All-Digital I/O Library Cells for System-in-Package Integration of Multiple Dies"*, IEEE 27$^{\text{th}}$ Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2018.

15. **H. M. Torun** and M. Swaminathan, *"Bayesian Framework for Optimization of Electromagnetics Problems"*, 2018 International Workshop on Computing, Electromagnetics, and Machine Intelligence (CEMi), Stellenbosch, 2018.

16. K. Roy, **H. M. Torun** and M. Swaminathan, *"Preliminary Application of Deep Learning to Design Space Exploration"*, IEEE Electrical Design of Advanced Packaging & Systems Symposium, 2018.

17. **H. M. Torun**, M. Larbi and M. Swaminathan, *"A Bayesian Framework for Optimizing Interconnects in High-Speed Channels"*, IEEE MTT-S Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO), 2018.

18. M. Larbi, **H. M. Torun**, M. Swaminathan, I. S. Stievano, F. G. Canevero and P. Besnier *"Uncertainty Quantification of SiP based Integrated Voltage Regulator"*, 22nd IEEE Workshop on Signal and Power Integrity (SPI) 2018.

19. C. Pardue, **H. M. Torun**, M. Bellaredj and M. Swaminathan, *"System Level Efficiency Analysis for Regulated RF Near Field Coupling"*, IEEE MTT-S Wireless Power Transfer Conference (WPTC) 2018.

20. E. Lee, M. Amir, S. Sivapurapu, C. Pardue, **H. M. Torun**, M. Bellaredj, M. Swaminathan and S. Mukhopadyay *"A System-in-Package Based Energy Harvesting for IoT Devices with Integrated Voltage Regulators and Embedded Inductors"*, IEEE 68th Electron. Compon. Technol. Conf. (ECTC) 2018.

21. **H. M. Torun**, C. Pardue, M. L. F Belladredj, A. K. Davis and M. Swaminathan, *"Machine Learning Driven Advanced Packaging and System Miniaturization of IoT for Wireless Power Transfer Solutions"*, IEEE 68th Electron. Compon. Technol. Conf. (ECTC) 2018.

22. **H. M. Torun** and M. Swaminathan, *"A New Machine Learning Approach for Optimization and Tuning of Integrated Systems"*, DesignCon 2018.

23. **H. M. Torun** and M. Swaminathan, *"Black-Box Optimization of 3D Integrated Circuits using Machine Learning"*, IEEE 26th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS), 2017.

24. **H. M. Torun** and M. Swaminathan, *"Black-Box Optimization of 3D Integrated Circuits"*, Computational Modelling of Multi-Uncertainty and Multi-Scale Problems (COMUS), 2017.

9.3 Future Work

Continuous development of new techniques have the potential to completely automate hardware design cycle in the future. Consider a typical design cycle that starts with some design objectives as in Figure 9.1. This is then followed by a design space exploration to find an initial design, which is then optimized and converted to layout for validation. During the design cycle, we need to iterate multiple times between DSE, optimization and validation as shown as loops in Figure 9.1. This takes considerable amount of time both due to computations and manual human intervention. The "human in the loop" is important for making design decisions and ensuring accuracy, but results in increased computations and delayed design closure. In addition, humans are prone to making mistakes and as a result, design re-spins cause increased delays. Instead, we believe that ML can remove the human from the loop as shown in Figure 9.1, make intelligent design decisions, generate optimized designs and eliminate errors, leading to significantly reduced design closure time.

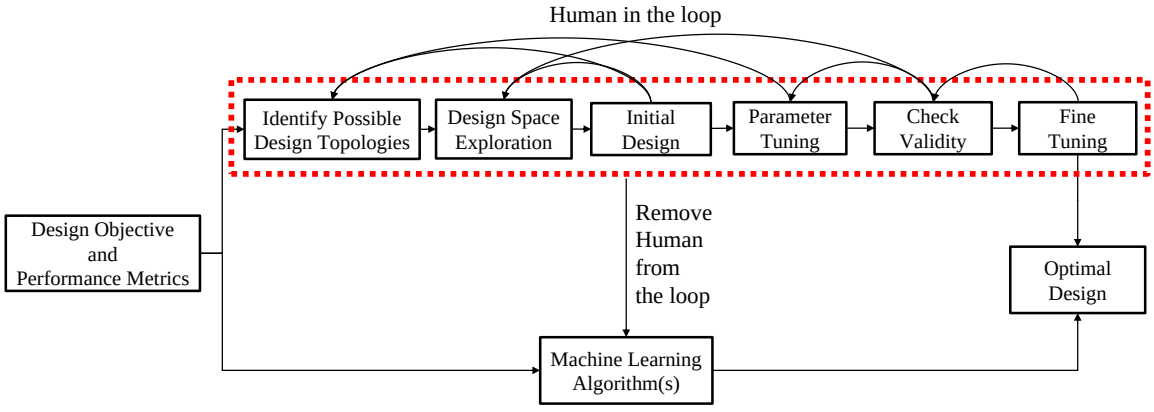


Figure 9.1: Design cycle time reduction and automation using ML.

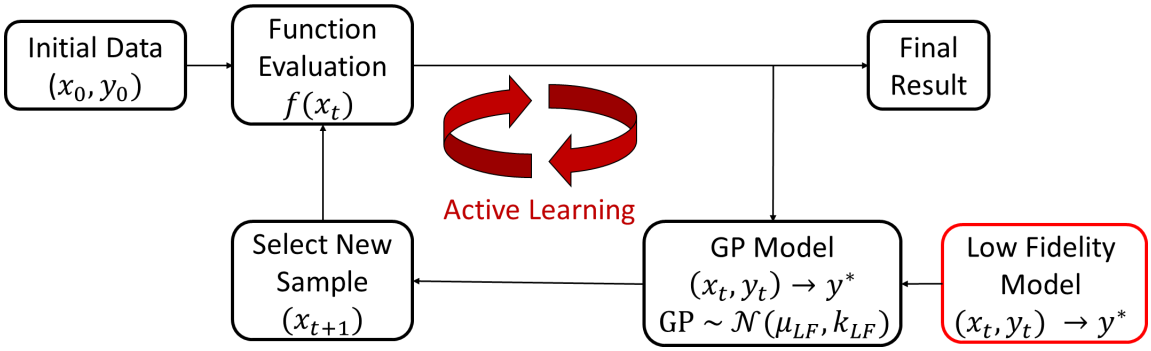


Figure 9.2: Multi-fidelity Bayesian Optimization framework.

Although this thesis provides a variety of techniques to eliminate manual human intervention and reduce design cycle time (for DSE, optimization and UQ), new techniques need to be developed in order to achieve the end goal of removing human from the design loop.

An example technique that can be immediately utilized is multi-fidelity design optimization. For many types of designs, it is possible to construct an approximate model that captures certain correlations that exist in the actual data. Such approximations include physics-based models, analytical equations or equivalent circuits that allow to collect vast amount of data in an efficient manner. The information contained in the data generated through such low-fidelity approximations can be fused with the data from CPU intensive high-fidelity simulations to reduce the total data collection time while maintaining a certain predictive accuracy. Existing methods in literature to perform such information fusion either assumes linear correlation between low and high-fidelity data, limited to low-fidelity data that uses the exact same input parameters, or can only capture stationary correlations. In this regard, it is possible to develop a multi-fidelity GP model where we fuse the low-fidelity information at the kernel level of our high-fidelity GP as in Figure 9.2. By introducing non-stationarity into kernel function, it is possible to create an expressive model that captures complex relationship between low-fidelity and high-fidelity data, hence, significantly improve the quality of the overall GP model.

REFERENCES

[1] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035.

[2] "Deep learning toolbox," *MATLAB*,

[3] W. T. Beyene, "Application of Artificial Neural Networks to Statistical Analysis and Nonlinear Modeling of High-Speed Interconnect Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, no. 1, Jan. 2007.

[4] Q.-J. Zhang and L. Zhang, "Neural Network Techniques for High-Speed Electronic Component Modeling," in *2009 IEEE MTT-S International Microwave Workshop Series on Signal Integrity and High-Speed Interconnects*, Feb. 2009.

[5] C. M. Schierholz, K. Scharff, and C. Schuster, "Evaluation of Neural Networks to Predict Target Impedance Violations of Power Delivery Networks," in *2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2019.

[6] T. Lu, J. Sun, K. Wu, and Z. Yang, "High-Speed Channel Modeling With Machine Learning Methods for Signal Integrity Analysis," *IEEE Transactions on Electromagnetic Compatibility*, no. 6, Dec. 2018.

[7] C. H. Goay, A. Abd Aziz, N. S. Ahmad, and P. Goh, "Eye Diagram Contour Modeling Using Multilayer Perceptron Neural Networks With Adaptive Sampling and Feature Selection," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, no. 12, Dec. 2019.

[8] S. Chen, J. Chen, T. Zhang, and S. Wei, "Semi-supervised learning based on Hybrid Neural Network for the Signal Integrity Analysis," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2019.

[9] Y. Liu, T. Lu, J. Y. Kim, K. Wu, and J.-M. Jin, "Fast and Accurate Current Prediction in Packages Using Neural Networks," in *2019 IEEE International Symposium on Electromagnetic Compatibility, Signal Power Integrity (EMC+SIPI)*, Jul. 2019.

[10] B. Mutnury, M. Swaminathan, M. Cases, N. Pham, D. de Araujo, and E. Matoglu, "Macro-modeling of non-linear pre-emphasis differential driver circuits," in *IEEE MTT-S International Microwave Symposium Digest, 2005.*, Jun. 2005.

[11] Q. Zhang, Y. Cao, and I. Erdin, "Fast IO buffer modeling using neural network methods," in *2010 11th International Conference on Electronic Packaging Technology High Density Packaging*, Aug. 2010.

[12] T. Nguyen, T. Lu, J. Sun, Q. Le, K. We, and J. Schut-Aine, "Transient Simulation for High-Speed Channels with Recurrent Neural Network," in *2018 IEEE 27th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2018.

[13] T. Nguyen, X. Wang, X. Chen, and J. Schutt-Aine, "A Deep Learning Approach for Volterra Kernel Extraction for Time Domain Simulation of Weakly Nonlinear Circuits," in *2019 IEEE 69th Electronic Components and Technology Conference (ECTC)*, May 2019.

[14] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[15] A. G. Wilson, "Covariance kernels for fast automatic pattern discovery and extrapolation with gaussian processes," *University of Cambridge*, 2014.

[16] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in neural information processing systems*, 2012, pp. 2951–2959.

[17] D. Duvenaud, "Automatic model construction with gaussian processes," PhD thesis, University of Cambridge, 2014.

[18] F. Feng, C. Zhang, S. Zhang, V. Gongal-Reddy, and Q. Zhang, "Parallel EM optimization approach to microwave filter design using feature assisted neuro-transfer functions," in *2016 IEEE MTT-S International Microwave Symposium (IMS)*, May 2016, pp. 1–3.

[19] F. Feng, C. Zhang, J. Ma, and Q. Zhang, "Parametric modeling of em behavior of microwave components using combined neural networks and pole-residue-based transfer functions," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 1, pp. 60–77, Jan. 2016.

[20] A. Ciccazzo, G. D. Pillo, and V. Latorre, "A SVM Surrogate Model-Based Method for Parametric Yield Optimization," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 7, pp. 1224–1228, Jul. 2016.

[21] F. Mokhupuki and D. I. L. de Villiers, "Surrogate Based Optimization of Wideband Reflector Feed Antennas," in *2019 13th European Conference on Antennas and Propagation (EuCAP)*, Mar. 2019, pp. 1–5.

[22] T. Dhaene, "Challenges in the optimization of modern rf and em systems: A bayesian perspective," in *2018 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, 2018.

[23] J. T. Springenberg, A. Klein, S. Falkner, and F. Hutter, "Bayesian optimization with robust bayesian neural networks," in *Advances in Neural Information Processing Systems 29*, 2016, pp. 4134–4142.

[24] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International Conference on Learning and Intelligent Optimization*, Springer, 2011, pp. 507–523.

[25] J. S. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Advances in neural information processing systems*, 2011, pp. 2546–2554.

[26] M. Feurer, B. Letham, and E. Bakshy, "Scalable meta-learning for bayesian optimization," *CoRR*, vol. abs/1802.02219, 2018.

[27] R. M. Neal, *Bayesian learning for neural networks*. Springer Science & Business Media, 2012, vol. 118.

[28] C. E. Rasmussen and Z. Ghahramani, "Occam's razor," in *Advances in neural information processing systems*, 2001, pp. 294–300.

[29] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels," *Journal of Machine Learning Research*, vol. 7, pp. 2651–2667, Dec. 2006.

[30] B. Sriperumbudur, K. Fukumizu, and G. Lanckriet, "On the relation between universality, characteristic kernels and rkhs embedding of measures," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 773–780.

[31] H. J. Kushner, "A New Method of Locating the Maximum Point of an Arbitrary Multipeak Curve in the Presence of Noise," *Journal of Basic Engineering*, vol. 86, no. 1, pp. 97–106, Mar. 1964. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/86/1/97/5763745/97_1.pdf.

[32] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of bayesian methods for seeking the extremum," *Towards global optimization*, vol. 2, no. 117-129, p. 2, 1978.

[33] N. Srinivas, A. Krause, S. Kakade, and M. Seeger, "Gaussian process optimization in the bandit setting: No regret and experimental design," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, 2010, pp. 1015–1022, ISBN: 978-1-60558-907-7.

[34] H. Kabir, L. Zhang, M. Yu, P. H. Aaen, J. Wood, and Q.-J. Zhang, "Smart Modeling of Microwave Devices," *IEEE Microwave Magazine*, vol. 11, no. 3, pp. 105–118, May 2010.

[35] H. M. Torun, M. Larbi, and M. Swaminathan, "A Bayesian Framework for Optimizing Interconnects in High-Speed Channels," in *2018 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, Aug. 2018, pp. 1–4.

[36] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[37] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, Apr. 2017.

[38] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[39] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[40] F. Wang and Q.-J. Zhang, "Knowledge-based neural models for microwave design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no. 12, pp. 2333–2343, 1997.

[41] W. Na, F. Feng, C. Zhang, and Q.-J. Zhang, "A Unified Automated Parametric Modeling Algorithm Using Knowledge-Based Neural Network and l_1 Optimization," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 3, pp. 729–745, 2017.

[42] P. Triverio, "Robust Causality Check for Sampled Scattering Parameters via a Filtered Fourier Transform," *IEEE Microwave and Wireless Components Letters*, vol. 24, no. 2, pp. 72–74, Feb. 2014.

[43] P. Triverio, S. Grivet-Talocia, M. S. Nakhla, F. G. Canavero, and R. Achar, "Stability, causality, and passivity in electrical interconnect models," *IEEE Transactions on Advanced Packaging*, 2007.

[44] S. Grivet-Talocia, "Passivity enforcement via perturbation of Hamiltonian matrices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 9, pp. 1755–1769, Sep. 2004.

[45] L. L. Barannyk, H. A. Aboutaleb, A. Elshabini, and F. D. Barlow, "Spectrally Accurate Causality Enforcement Using SVD-Based Fourier Continuations for High-Speed Digital Interconnects," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 5, no. 7, pp. 991–1005, Jul. 2015.

[46] J. S. Toll, "Causality and the Dispersion Relation: Logical Foundations," *Physical Review*, vol. 104, no. 6, pp. 1760–1770, Dec. 1956.

[47] S. Grivet-Talocia, "Passivity enforcement via perturbation of hamiltonian matrices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 51, no. 9, pp. 1755–1769, Sep. 2004.

[48] A. Odena, V. Dumoulin, and C. Olah, "Deconvolution and Checkerboard Artifacts," *Distill*, vol. 1, no. 10, e3, Oct. 2016.

[49] A. Dienstfrey and L. Greengard, "Analytic continuation, singular-value expansions, and Kramers-Kronig analysis," *Inverse Problems*, vol. 17, no. 5, pp. 1307–1320, Oct. 2001.

[50] L. Marple, "Computing the discrete-time "analytic" signal via FFT," *IEEE Transactions on Signal Processing*, vol. 47, no. 9, pp. 2600–2603, Sept./1999.

[51] D. W. Robinson, "A Note on a Simple Matrix Isomorphism," *Mathematics Magazine*, vol. 32, no. 4, pp. 213–215, 1959.

[52] J. K. Merikoski, H. Sarria, and P. Tarazaga, "Bounds for singular values using traces," *Linear Algebra and its Applications*, vol. 210, pp. 227–254, 1994.

[53] H. Wolkowicz and G. P. Styan, "Bounds for eigenvalues using traces," *Linear Algebra and its Applications*, vol. 29, pp. 471–506, Feb. 1980.

[54] H. M. Torun, A. C. Durgun, K. Aygün, and M. Swaminathan, "Enforcing causality and passivity of neural network models of broadband s-parameters," in *2019 IEEE 28th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, Oct. 2019.

[55] K. Doshi, A. Sureka, and P. J. Pupalaikis, "Fast and Optimal Algorithms for Enforcing Reciprocity, Passivity and Causality in S-parameters," in *DesignCon*, 2012, p. 21.

[56] F. Tesche, "On the use of the Hilbert transform for processing measured CW data," *IEEE Transactions on Electromagnetic Compatibility*, vol. 34, no. 3, pp. 259–266, Aug. 1992.

[57] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus)*, 2015. arXiv: 1511.07289 [cs.LG].

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[59] Z. Kiguradze, J. He, B. Mutnury, A. Chada, and J. Drewniak, "Bayesian Optimization for Stack-up Design," in *2019 IEEE International Symposium on Electromagnetic Compatibility, Signal Power Integrity (EMC+SIPI)*, Jul. 2019.

[60] M. A. Dolatsara and M. Swaminathan, "Determining worst-case eye height in low BER channels using Bayesian optimization," in *2020 IEEE 11th Latin American Symposium on Circuits Systems (LASCAS)*, Feb. 2020.

[61] R. Medico, D. Spina, D. Vande Ginste, D. Deschrijver, and T. Dhaene, "Machine-Learning-Based Error Detection and Design Optimization in Signal Integrity Applications," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, no. 9, Sep. 2019.

[62] S. De Ridder, D. Spina, N. Toscani, F. Grassi, D. V. Ginste, and T. Dhaene, "Machine-Learning-Based Hybrid Random-Fuzzy Uncertainty Quantification for EMC and SI Assessment," *IEEE Transactions on Electromagnetic Compatibility*, 2020.

[63] K. Kawaguchi, L. P. Kaelbling, and T. Lozano-Pérez, "Bayesian optimization with exponential convergence," in *Advances in Neural Information Processing Systems*, 2015, pp. 2809–2817.

[64] Z. Wang, B. Shakibi, L. Jin, and N. Freitas, "Bayesian multi-scale optimistic optimization," in *Artificial Intelligence and Statistics*, 2014, pp. 1005–1014.

[65] R. Munos, "Optimistic optimization of a deterministic function without the knowledge of its smoothness," in *Advances in neural information processing systems*, 2011, pp. 783–791.

[66] M. Hoffman, E. Brochu, and N. de Freitas, "Portfolio allocation for bayesian optimization," in *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. UAI'11, 2011, pp. 327–336, ISBN: 978-0-9749039-7-2.

[67] B. Liu, H. Yang, and M. J. Lancaster, "Global optimization of microwave filters based on a surrogate model-assisted evolutionary algorithm," *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 6, pp. 1976–1985, Jun. 2017.

[68] S. Surjanovic, *Virtual library of simulation experiments:*

[69] D. J. Lizotte, "Practical bayesian optimization," *PhD Thesis, University of Alberta*, 2008.

[70] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of global optimization*, vol. 21, no. 4, pp. 345–383, 2001.

[71] E. Brochu, V. M. Cora, and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," arXiv.org, eprint arXiv:1012.2599, Dec. 2010.

[72] J. Xie and M. Swaminathan, "Electrical-thermal co-simulation of 3d integrated systems with micro-fluidic cooling and joule heating effects," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 1, no. 2, pp. 234–246, 2011.

[73] S. J. Park, B. Bae, J. Kim, and M. Swaminathan, "Application of machine learning for optimization of 3-d integrated circuits and systems," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[74] S. Mueller, K. Z. Ahmed, A. Singh, A. K. Davis, S. Mukhopadyay, M. Swaminathan, Y. Mano, Y. Wang, J. Wong, S. Bharathi, H. F. Moghadam, and D. Draper, "Design of high efficiency integrated voltage regulators with embedded magnetic core inductors," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, May 2016, pp. 566–573.

[75] D. S. Gardner, G. Schrom, F. Paillet, B. Jamieson, T. Karnik, and S. Borkar, "Review of on-chip inductor structures with magnetic films," *IEEE Transactions on Magnetics*, vol. 45, no. 10, pp. 4760–4766, Oct. 2009.

[76] M. L. F. Bellaredj, S. Mueller, A. K. Davis, P. Kohl, M. Swaminathan, and Y. Mano, "Fabrication, characterization and comparison of fr4-compatible composite magnetic materials for high efficiency integrated voltage regulators with embedded magnetic core micro-inductors," in *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*, 2017, pp. 2008–2014.

[77] K. Kandasamy, J. Schneider, and B. Póczos, "High dimensional bayesian optimisation and bandits via additive models," in *International Conference on Machine Learning*, 2015, pp. 295–304.

[78] Z. Wang, C. Li, S. Jegelka, and P. Kohli, "Batched high-dimensional Bayesian optimization via structural kernel learning," in *Proceedings of the 34th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Aug. 2017, pp. 3656–3664.

[79] C.-L. Li, K. Kandasamy, B. Poczos, and J. Schneider, "High dimensional bayesian optimization via restricted projection pursuit models," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 51, PMLR, May 2016, pp. 884–892.

[80] D. K. Duvenaud, H. Nickisch, and C. E. Rasmussen, "Additive gaussian processes," in *Advances in Neural Information Processing Systems 24*, 2011, pp. 226–234.

[81] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas, "Bayesian optimization in high dimensions via random embeddings," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13, Beijing, China, 2013, pp. 1778–1784, ISBN: 978-1-57735-633-2.

[82] J. Djolonga, A. Krause, and V. Cevher, "High-dimensional gaussian process bandits," in *Advances in Neural Information Processing Systems*, 2013, pp. 1025–1033.

[83] G. Li, C. Rosenthal, and H. Rabitz, "High dimensional model representations," *The Journal of Physical Chemistry A*, vol. 105, no. 33, pp. 7765–7777, 2001. eprint: https://doi.org/10.1021/jp010450t.

[84] J. M. Hernández-Lobato, M. W. Hoffman, and Z. Ghahramani, "Predictive entropy search for efficient global optimization of black-box functions," in *Advances in neural information processing systems*, 2014, pp. 918–926.

[85] Z. Wang and S. Jegelka, "Max-value entropy search for efficient bayesian optimization," in *International Conference on Machine Learning (ICML)*, 2017.

[86] H. M. Torun, M. Larbi, and M. Swaminathan, "A bayesian framework for optimizing interconnects in high-speed channels," in *2018 IEEE MTT-S International Conference on Numerical Electromagnetic and Multiphysics Modeling and Optimization (NEMO)*, Aug. 2018, pp. 1–4.

[87] F. Parment, A. Ghiotto, T.-P. Vuong, J.-M. Duchamp, and K. Wu, "Air-filled substrate integrated waveguide for low-loss and high power-handling millimeter-wave substrate integrated circuits," *IEEE Transactions on Microwave Theory and Techniques*, vol. 63, no. 4, pp. 1228–1238, 2015.

[88] M. Yi, S. Li, H. Yu, W. Khan, C. Ulusoy, A. Vera-Lopez, J. Papapolymerou, and M. Swaminathan, "Surface roughness modeling of substrate integrated waveguide in d-band," *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 4, pp. 1209–1216, Apr. 2016.

[89] C. Pardue, A. Davis, M. Bellaredj, and M. Swaminathan, "Wireless power transfer integrated board for low power iot applications," in *2017 IEEE Conference on Technologies for Sustainability (SusTech)*, Nov. 2017, pp. 1–6.

[90] H. M. Torun, C. Pardue, M. L. F. Belleradj, A. K. Davis, and M. Swaminathan, "Machine learning driven advanced packaging and miniaturization of iot for wireless power transfer solutions," in *2018 IEEE 68th Electronic Components and Technology Conference (ECTC)*, May 2018, pp. 2374–2381.

[91] Linear Technology, *Ltc3564 datasheet*.

[92] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley, "Google Vizier: A Service for Black-Box Optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17, New York, NY, USA: Association for Computing Machinery, Aug. 2017, ISBN: 978-1-4503-4887-4.

[93] J. Snoek, K. Swersky, R. Zemel, and R. Adams, "Input warping for bayesian optimization of non-stationary functions," in *International Conference on Machine Learning*, 2014, pp. 1674–1682.

[94] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*, 2016, pp. 370–378.

[95] J. Blank and K. Deb, "Pymoo: Multi-objective optimization in python," *IEEE Access*, pp. 1–1, 2020.

[96] T. Head, MechCoder, G. Louppe, I. Shcherbatyi, fcharras, Z. Vinícius, cmmalone, C. Schröder, nel215, N. Campos, T. Young, S. Cereda, T. Fan, J. Schwabedal, Hvass-Labs, M. Pak, SoManyUsernamesTaken, F. Callaway, L. Estève, L. Besson, P. M. Landwehr, P. Komarov, M. Cherti, K. (Shi, K. Pfannschmidt, F. Linzberger, C. Cauet, A. Gut, A. Mueller, and A. Fabisch, *scikit-optimize/scikit-optimize: v0.5.1 - re- release*, version v0.5.1, Feb. 2018.

[97] B. Ru, A. S. Alvi, V. Nguyen, M. A. Osborne, and S. J. Roberts, "Bayesian Optimisation over Multiple Continuous and Categorical Inputs," in *37th International Conference on Machine Learning (ICML)*, 2020.

[98] M. T. M. Emmerich and A. H. Deutz, "A tutorial on multiobjective optimization: Fundamentals and evolutionary methods," *Natural Computing*, no. 3, Sep. 2018.

[99] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, no. 1, Feb. 2006.

[100] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection," in *International Conference on Parallel Problem Solving from Nature*, Springer, 2008, pp. 784–794.

[101] I. Couckuyt, D. Deschrijver, and T. Dhaene, "Fast calculation of multiobjective probability of improvement and expected improvement criteria for pareto optimization," *Journal of Global Optimization*, vol. 60, no. 3, pp. 575–594, 2014.

[102] A. Iyer, Y. Zhang, A. Prasad, S. Tao, Y. Wang, L. Schadler, L. C. Brinson, and W. Chen, "Data-centric mixed-variable bayesian optimization for materials design," in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, vol. 59186, 2019, V02AT03A066.

[103] S. Suzuki, S. Takeno, T. Tamura, K. Shitara, and M. Karasuyama, "Multi-objective bayesian optimization using pareto-frontier entropy," *arXiv preprint arXiv:1906.00127*, 2019.

[104] S. Belakaria, A. Deshwal, and J. R. Doppa, "Max-value entropy search for multi-objective bayesian optimization," in *Advances in Neural Information Processing Systems*, 2019, pp. 7825–7835.

[105] K. Kandasamy, K. R. Vysyaraju, W. Neiswanger, B. Paria, C. R. Collins, J. Schneider, B. Poczos, and E. P. Xing, "Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly," *Journal of Machine Learning Research*, vol. 21, no. 81, pp. 1–27, 2020.

[106] S. Nozawa, T. Maekawa, E. Yagi, Y. Terao, and H. Kohno, "Development of new power control unit for compact-class vehicle," in *2010 22nd International Symposium on Power Semiconductor Devices IC's (ISPSD)*, Jun. 2010.

[107] H. Lee, V. Smet, and R. Tummala, "A Review of SiC Power Module Packaging Technologies: Challenges, Advances, and Emerging Issues," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, no. 1, Mar. 2020.

[108] A. Ibrahim, S. Rahnamayan, M. V. Martin, and K. Deb, "3D-RadVis: Visualization of Pareto front in many-objective optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Jul. 2016.

[109] T. A. Burress, C. Coomer, S. Campbell, A. Wereszczak, J. Cunningham, L. Marlino, L. Seiber, and H.-T. Lin, "Evaluation of the 2008 lexus ls 600h hybrid synergy drive system," Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, Tech. Rep., 2009.

[110] R. Trinchero, M. Larbi, H. M. Torun, F. G. Canavero, and M. Swaminathan, "Machine learning and uncertainty quantification for surrogate models of integrated devices with a large number of parameters," *IEEE Access*, vol. 7, pp. 4056–4066, 2018.

[111] R. M. Neal, "Slice sampling," *Annals of statistics*, pp. 705–741, 2003.

[112] J. Vanhatalo, J. Riihimäki, J. Hartikainen, P. Jylänki, V. Tolvanen, and A. Vehtari, "Gpstuff: Bayesian modeling with gaussian processes," *Journal of Machine Learning Research*, vol. 14, no. Apr, pp. 1175–1179, 2013.

[113] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *Journal of Machine Learning Research*, vol. 9, no. Feb, pp. 235–284, 2008.

[114] S. Chun, W. D. Becker, J. Casey, S. Ostrander, D. Dreps, J. A. Hejase, R. M. Nett, B. Beaman, and J. R. Eagle, "Ibm power9 package technology and design," *IBM Journal of Research and Development*, vol. 62, no. 4/5, 12:1–12:10, Jul. 2018.

[115] H. M. Torun, M. Larbi, and M. Swaminathan, "Machine learning-based system optimization and uncertainty quantification for integrated systems," in *Machine Learning in VLSI Computer-Aided Design*, Springer, 2019, pp. 505–536.

[116] J. Kim, G. Murali, H. Park, E. Qin, H. Kwon, V. C. K. Chekuri, N. M. Rahman, N. Dasari, A. Singh, M. Lee, H. M. Torun, K. Roy, M. Swaminathan, S. Mukhopadhyay, T. Krishna, and S. K. Lim, "Architecture, chip, and package codesign flow for interposer-based 2.5-d chiplet integration enabling heterogeneous ip reuse," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 11, pp. 2424–2437, 2020.

[117] M. Swaminathan, H. M. Torun, H. Yu, J. A. Hejase, and W. D. Becker, "Demystifying machine learning for signal and power integrity problems in packaging," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 10, no. 8, pp. 1276–1295, 2020.

[118] H. M. Torun, A. C. Durgun, K. Aygün, and M. Swaminathan, "Causal and passive parameterization of s-parameters using neural networks," *IEEE Transactions on Microwave Theory and Techniques*, vol. 68, no. 10, pp. 4290–4304, 2020.

[119] S. Mukhopadhyay, Y. Long, B. Mudassar, C. Nair, B. H. DeProspo, H. M. Torun, M. Kathaperumal, V. Smet, D. Kim, S. Yalamanchili, *et al.*, "Heterogeneous integration for artificial intelligence: Challenges and opportunities," *IBM Journal of Research and Development*, vol. 63, no. 6, pp. 4–1, 2019.

[120] H. M. Torun and M. Swaminathan, "High-dimensional global optimization method for high-frequency electronic design," *IEEE Transactions on Microwave Theory and Techniques*, vol. 67, no. 6, pp. 2128–2142, Jun. 2019.

[121] C. A. Pardue, H. M. Torun, M. L. F. Bellaredj, and M. Swaminathan, "Rf near-field coupling system using reverse pdn and considering electromagnetic effects,"

IEEE Transactions on Electromagnetic Compatibility, vol. 61, no. 6, pp. 1904–1912, 2019.

[122] C. A. Pardue, M. L. F. Bellaredj, H. M. Torun, M. Swaminathan, P. Kohl, and A. K. Davis, "Rf wireless power transfer using integrated inductor," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 9, no. 5, pp. 913–920, 2018.

[123] H. M. Torun, M. Swaminathan, A. Kavungal Davis, and M. L. F. Bellaredj, "A global bayesian optimization algorithm and its application to integrated system design," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 4, pp. 792–802, Apr. 2018.

[124] X. Yang, J. Tang, H. M. Torun, W. D. Becker, J. A. Hejase, and M. Swaminathan, "Rx equalization for a high-speed channel based on bayesian active learning using dropout," in *2020 IEEE 29th Conference on Electrical Performance of Electronic Packaging and Systems (EPEPS)*, IEEE, 2020, pp. 1–3.

[125] K. Scharff, H. M. Torun, C. Yang, M. Swaminathan, and C. Schuster, "Bayesian optimization for signal transmission including crosstalk in a via array," in *2020 International Symposium on Electromagnetic Compatibility-EMC EUROPE*, IEEE, 2020, pp. 1–6.

[126] H. Yu, H. M. Torun, M. U. Rehman, and M. Swaminathan, "Design of siw filters in d-band using invertible neural nets," in *2020 IEEE/MTT-S International Microwave Symposium (IMS)*, IEEE, 2020, pp. 72–75.